MULTI-LANGUAGE TEXT EXTRACTION AND IDENTIFICATION

A report on Project Phase – I submitted

in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

by

Zayiechutuo Lohe
Registration No.: 2020105263

Yuvraj Kumar
Registration No.: 2020105262

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY NAGALAND

NOVEMBER 2023

NATIONAL INSTITUTE OF TECHNOLOGY NAGALAND
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Chumukedima, Dimapur – 797 103, Nagaland

BONAFIDE CERTIFICATE

Certified that this Project titled **"MULTI-LANGUAGE TEXT EXTRACTION AND IDENTIFICATION"** is the bonafide work of **Zayiechutuo Lohe (2020105263) & Yuvraj Kumar (2020105262)** who carried out the work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other student.

Dr. ARAMBAM NEELIMA
Project Coordinator
Assistant Professor
Department of computer Science
and Engineering
NIT Nagaland
Dimapur – 797 103

Dr. Shouvik Dey
Project Supervisor
Assistant Professor
Department of Computer Science
and Engineering
NIT Nagaland
Dimapur – 797 103

Dr. LITHUNGO MURRY
Head of Department
Assistant Professor
Department of computer Science
and Engineering
NIT Nagaland
Dimapur – 797 103

Dr. H. ROHEN SINGH
External Examiner
Assistant Professor
Department of IT
School of Engineering and Technology
Nagaland University
Dimapur – 797 103

ABSTRACT

The project work reported on the development of "Extraction and Identification of text from images" through the integration of Optical Character Recognition (OCR) and advanced image preprocessing techniques. The primary objective was to enhance the accuracy and efficiency of text extraction from diverse image sources.

The implementation involved the utilization of cutting-edge OCR algorithms to recognize and transcribe text elements from images, coupled with rigorous image preprocessing methods to optimize data quality. Various image enhancement techniques, such as noise reduction, contrast adjustment, and morphological operations, were employed to ensure robust performance. Notably, the system demonstrated a proficiency in identifying text in up to 17 languages, showcasing its versatility across multilingual contexts.

The synergy of OCR and image preprocessing not only facilitated the extraction process but also significantly improved the overall precision of text identification. The outcomes of this research contribute to the advancement of text extraction methodologies, particularly in scenarios where image quality and diversity pose challenges to conventional extraction techniques.

ACKNOWLEDGEMENT

We wish to place on record our deep sense of gratitude to our honorific Guide **Dr. Shouvik Dey**, Designation, Institute for his supervision, valuable guidance and moral support leading to the successful completion of the work. Without his continuous encouragement and involvement, this project would not have been a reality.

We would like to extent my sincere thanks to **our friends** for their valuable suggestions. We wish to thank **Dr. Lithungo Murry**, Department of Computer Science and Engineering, NIT Nagaland for continuous support. We would also like to thank all

our friends who have developed us to gain a sense of dutifulness, perfection and sincerity in the effort.

We wish to dedicate this work to **our parents and teachers**, for they are the pillars of support giving us confidence in whatever we do. We would like to thank **Dr. Shouvik Dey** who has motivated us to work harder and do our best. Last but not least, We would like to owe our sincere and incessant gratitude to the almighty God for the immense blessing on us.

Zayiechutuo Lohe

&

Yuvraj Kumar

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

OCR - Optical Character Recognition

NLP - Natural Language Processing

ML - Machine Learning

AI - Artificial Intelligence

CV - Computer Vision

UI - User Interface

EDA - Exploratory Data Analysis

SVM - Support Vector Machine

MIP - Morphological Image Processing

KNN - K-Nearest Neighbors

CNN - Convolutional Neural Network

RNN - Recurrent Neural Network

TF-IDF - Term Frequency Inverse Document Frequency

DPI - Dots per Inch

PPI - Pixels per Inch

F-SCORE - A Measure That Combines Precision And Recall

NLTK - Natural Language Toolkit

CHAPTER 1

INTRODUCTION

1.1 General

Exploring the realm of extracting text from images, the project titled "Extraction and Identification of Text from Images" stands out by combining advanced Optical Character Recognition (OCR) and image preprocessing techniques. This method focuses on improving precision and adaptability, effectively transcribing text from various image sources. What's notable is its ability to identify content in up to 17 languages, showcasing versatility. By seamlessly integrating OCR and careful image preprocessing, the project emerges as a leader, overcoming challenges presented by diverse image qualities and different languages. This initiative sets new standards for accuracy and efficiency in the evolving landscape of text extraction methodologies.

1.2 State of the Art

In this section, a literature survey of related works on extraction of text from images and identification of languages from text is presented:

Rishabh Mittal and Anchal Garg [1] reviewed OCR where they discussed the use of OCR to extract texts from sources like handwritten documents or printed texts. They also discussed the use of models like SVM, KNN and PNN with accuracy score averaging around 89%. It was also found that Chinese and Arabic characters were particularly hard to identify.

Bishar Ahmed [2] discusses about Language Identification. Given that a content is given in a document, he proposed a model that can automatically identify the language of that given document using Naïve Bayesian Classification with accuracy

of 80%. The issue with his methodologies were that they were unable to provide consistent results in slanted texts or texts covered with content lines.

Vicky Kumar [3] wrote on language identification by using various machine learning models and pattern recognition to provide insights and enhance various aspects of communication, interaction and analysis of a spoken language. However the model was only tested in 3 languages namely English, Hindi and Tibetian.

Tom Kocmi [4] published on multilingual language identification on character window. It used characters from multiple languages and determining it's language by understanding the pattern and letters of those languages. He used Bi-directional Neural Network to cover highly accurate identification of 3 languages. He used 6 datasets to cover those 3 languages and his model works well on 3 languages.

1.3 Objectives

The main objectives of this project are to extract text from images and translate them into English if not already in English. This can be achieved by using various Image preprocessing strategies to clean the image of any noise or enhance degraded images and after that using OCR to extract texts from those images. After the extraction of text has been completed, the identification of those extracted texts continues and if those identified language id not in English then translate the text to English.

1.4 Organization of the report

The organization of this thesis is as follows:

Chapter 1: This chapter will introduce the project, objectives and literature survey.

Chapter 2: This chapter discusses the data collection methods and the different machine learning models that were used for language identification along with the flow diagram and also it introduces all the materials and libraries used.

Chapter 3: This chapter will describe the steps that were used during image preprocessing.

Chapter 4: This chapter will enumerate how text extraction and identification works in this project

Chapter 5: This chapter will give the detail of the result and comparative analysis of the model based on different matrices.

Chapter 6: This chapter includes conclusion, work done, future work and references.

CHAPTER 2

DATA COLLECTION AND PROPOSED METHODOLOGY

2.1 Introduction

This section will involve explaining the various materials, Libraries and Machine learning Models that we have tested for this project.

2.2 Dataset

It's a small language detection dataset. This dataset consists of text details for 17 different languages, ie, English, Malayalam, Hindi, Tamil, Kannada, French, Spanish, Portuguese, Italian, Russian, Sweedish, Dutch, Arabic, Turkish, German, Danish and Greek.

2.3 Materials and Libraries

This section will mention the various materials and libraries that were used and tested for the sake of this project.

2.3.1 Materials

JUPYTER NOTEBOOK: Jupyter Notebook is an open-source interactive web application that allows users to create and share live code, equations, visualizations, and narrative text. Supporting multiple programming languages, it facilitates data analysis, machine learning, and scientific research through an easy-to-use interface combining code and documentation in a single environment.

GIT: Git is a distributed version control system that enables collaborative software development. It tracks changes in source code during development, allowing multiple developers to work on a project simultaneously. Git provides features for branching, merging, and version history, enhancing code collaboration, tracking, and maintaining project integrity. It is widely used for

managing and coordinating code among teams, ensuring efficient development and version management.

2.3.2 Libraries

TESSERACT: It is an open-source Optical Character Recognition (OCR) engine developed by Google. It is designed to recognize text in images and convert it into machine-readable text. Tesseract supports multiple languages and provides accurate and efficient text extraction from various sources, making it a valuable tool for tasks such as document scanning, text recognition in images, and automated data extraction from documents. Tesseract's versatility, combined with ongoing community contributions, has made it a widely used and respected OCR solution in both research and practical applications.

OPENCV: Open Source Computer Vision Library, is a widely-used open-source computer vision and machine learning software library. Developed in C++ and with interfaces for Python and other languages, OpenCV provides a comprehensive set of tools for image and video processing, including functionalities for image manipulation, feature detection, object recognition, and machine learning. Its versatility makes it valuable in various applications such as robotics, augmented reality, facial recognition, and medical image analysis. OpenCV's active community and continuous development contribute to its status as a go-to resource for computer vision projects and research.

PILLOW(PIL): Pillow, often referred to as PIL (Python Imaging Library), is an open-source Python library for image processing tasks. It provides a user-friendly interface for opening, manipulating, and saving various image file formats. Pillow supports basic image operations such as cropping, resizing, and filtering, making it a versatile tool for tasks like graphics processing, web development, and data analysis. As a fork of the original PIL, Pillow is actively maintained, ensuring compatibility with modern Python versions and continued support for image-related tasks in diverse applications.

NUMPY: Numerical Python, is a powerful open-source library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy is a fundamental tool in scientific computing, machine learning, and data analysis, offering capabilities for numerical operations, linear algebra, statistical analysis, and more. Its efficient array operations and broadcasting functionality enhance performance and code readability. NumPy serves as the foundation for many other scientific computing libraries in the Python ecosystem, making it an essential component for numerical work in Python.

PANDAS: Pandas is an open-source data manipulation and analysis library for Python. It provides easy-to-use data structures, such as DataFrame and Series, for efficiently handling and analyzing structured data. With powerful tools for cleaning, transforming, and aggregating data, Pandas is widely used in data science, statistics, and finance. It simplifies tasks like data cleaning, exploration, and manipulation, allowing users to work with labeled and relational data seamlessly. Pandas also integrates well with other libraries, making it a key component in the Python data science ecosystem, enabling researchers and analysts to extract valuable insights from diverse datasets.

MATPLOTLIB: Matplotlib is a versatile open-source library for creating static, animated, and interactive visualizations in Python. It provides a wide range of plotting functions to generate diverse types of charts, graphs, and plots. With customizable features, Matplotlib is extensively used in data visualization, scientific research, and educational presentations. It supports various plot styles, including line plots, scatter plots, bar plots, and more. Matplotlib's compatibility with NumPy arrays and Pandas DataFrames makes it a powerful tool for representing data visually and conveying complex information in a clear and

effective manner. It is a fundamental library in the Python data visualization ecosystem.

SEABORN: Seaborn is a statistical data visualization library in Python, built on top of Matplotlib. It provides a high-level interface for creating aesthetically pleasing and informative statistical graphics. Seaborn simplifies the process of generating complex visualizations by offering functions for common statistical plots, such as scatter plots, box plots, and histograms, with enhanced color palettes and styling.

SCIKIT-LEARN: Scikit-learn is a powerful and widely used open-source machine learning library for Python. It provides simple and efficient tools for data analysis and modeling, offering a comprehensive range of machine learning algorithms for tasks such as classification, regression, clustering, and dimensionality reduction. Scikit-learn is designed to work seamlessly with other scientific computing libraries like NumPy, SciPy, and Matplotlib.

NLTK: NLTK, or the Natural Language Toolkit, is a comprehensive open-source library for natural language processing (NLP) in Python. It provides tools and resources for working with human language data, including extensive corpora, lexical resources, and various text processing modules. NLTK supports a wide range of NLP tasks, such as tokenization, stemming, part-of-speech tagging, parsing, and sentiment analysis.

STREAMLIT: Streamlit is an open-source Python library that simplifies the process of creating web applications for data science and machine learning projects. With a straightforward and user-friendly syntax, Streamlit enables developers and data scientists to turn data scripts into interactive web apps quickly. It supports features like widgets for user interaction, real-time updates, and integration with popular data science libraries like Pandas, Matplotlib, and Plotly.

TENSORFLOW: TensorFlow is an open-source machine learning framework by Google, supporting deep learning and traditional ML. It offers flexibility, scalability, and high-level APIs like Keras. With TensorBoard for visualization, it's widely used in research and industry for applications such as image recognition and natural language processing.

2.4 PROPOSED MODELS

2.4.1 Design:

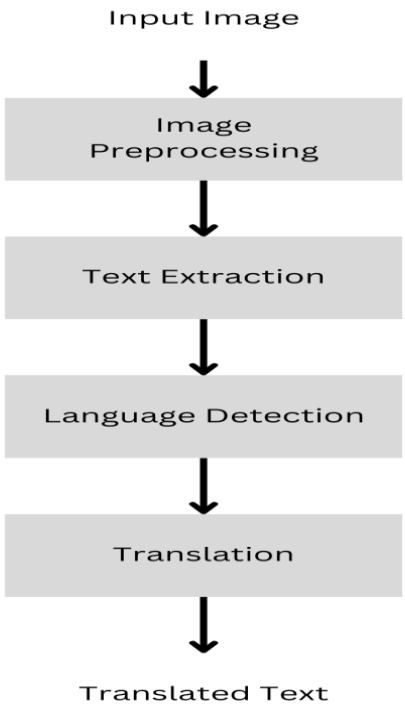


Fig 2.1 Proposed Methodology

This is the overall view of our project and the above flowchart describes how we plan to go forward with the project.

2.4.2 Logistic Regression:

This type of statistical model (also known as the *logit model*) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voting or didn’t vote, based on a given

dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, logit transformation is applied to the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds or the natural logarithm of odds, and this logistic function is represented by the following formulas:

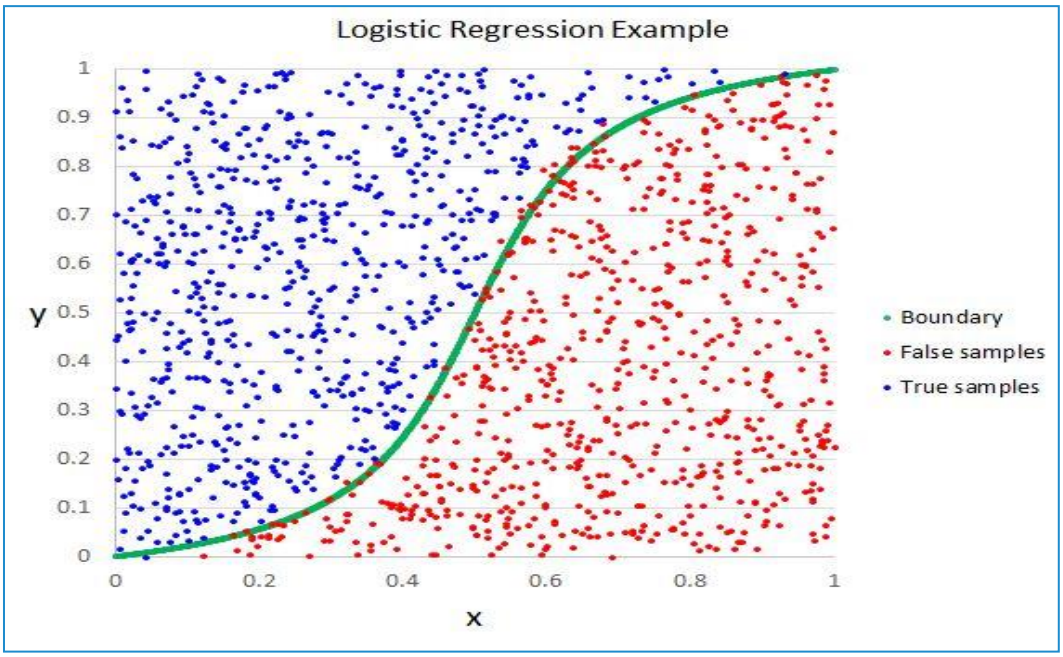
$$\text{Logit}(pi) = 1/ (1+ \exp(-pi)) \quad (1)$$

$$\ln(pi/(1-pi)) = \text{Beta}_0 + \text{Beta}_1*X_1 + \ldots + B_k*K_k \quad (2)$$

Logit(pi) is the dependent or response variable in this logistic regression equation whereas x is the independent variable. In this model, the beta parameter or coefficient is frequently determined using the maximum likelihood method (MLE). In order to get the best fit for the log odds, this approach iteratively evaluates various beta values. Logistic regression aims to maximize this function after each of these rounds in order to get the optimal parameter estimate. The conditional probabilities for each observation may be calculated, logged, and added together to provide a forecast probability after the best coefficient (or coefficients, if there are several independent variables) has been identified. A probability of less than.5 will predict 0 in a binary classification, while a probability of more than .5 will predict 1. After the model has been computed, it's best practice to evaluate how well the model predicts the dependent variable, which is called goodness of fit.

Logistic Regression can be of three types based on the categorical response:

- Binary Logistic Regression
- Multinomial Logistic Regression
- Ordinal Logistic Regression



https://www.voxco.com/wpcontent/uploads/2021/11/LOGISTIC-REGRESSION.jpg

Fig 2.2 Classification of news using Logistic Regression

2.4.3 Decision Tree:

DT is an important supervised learning algorithm. Researchers tend to use tree-based ensemble models like Random Forest or Gradient Boosting on all kinds of tasks. The basic idea of DT is that it develops a model to predict the value of a dependent factor by learning various decision rules inferred from the whole data. A decision Tree has a top-down structure and shapes like a tree in which a node can only be a leaf node that is bound with a label class or a decision node that is responsible for making decisions. A Decision Tree is easily understandable about the process of making decisions and predictions. However, it is a weak learner which means it may have bad performance on small datasets.

The key learning process in DT is to select the best attribute. To solve this problem, various trees have different metrics such as information gain used in the ID3 algorithm and gain ratio used in this algorithm. A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the formula below:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy (each feature)] (1)

Entropy(s)= - P(yes) log2 P(yes) - P(no) log2 P(no) (2)

Where,

- S= Total number of samples
- P(yes)= probability of yes
- P(no)= probability of no

2.4.4 Naïve Bayes:

The Nave Bayes algorithm is a supervised learning method for classification issues that is based on the Bayes theorem. It is mostly employed in text categorization with a large training set. The Naive Bayes Classifier is one of the most straightforward and efficient classification algorithms available today. It aids in the development of rapid machine learning models capable of making accurate predictions. Being a probabilistic classifier, it makes predictions based on the likelihood that an object will occur. The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

- Bayes: It is called Bayes because it depends on the principle of Bayes Theorem. The Bayes theorem, commonly referred to as Bayes' Rule or Bayes' law, is a formula for calculating the likelihood of a hypothesis in the presence of data. The conditional probability will determine what happens. The formula for Bayes' theorem is given as

$$P(A/B) = \frac{P\left(\frac{B}{A}\right) * P(A)}{P(B)} \qquad (1)$$

Where,

- P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.
- P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.
- P(A) is Prior Probability: Probability of hypothesis before observing the evidence.
- P(B) is Marginal Probability: Probability of Evidence.

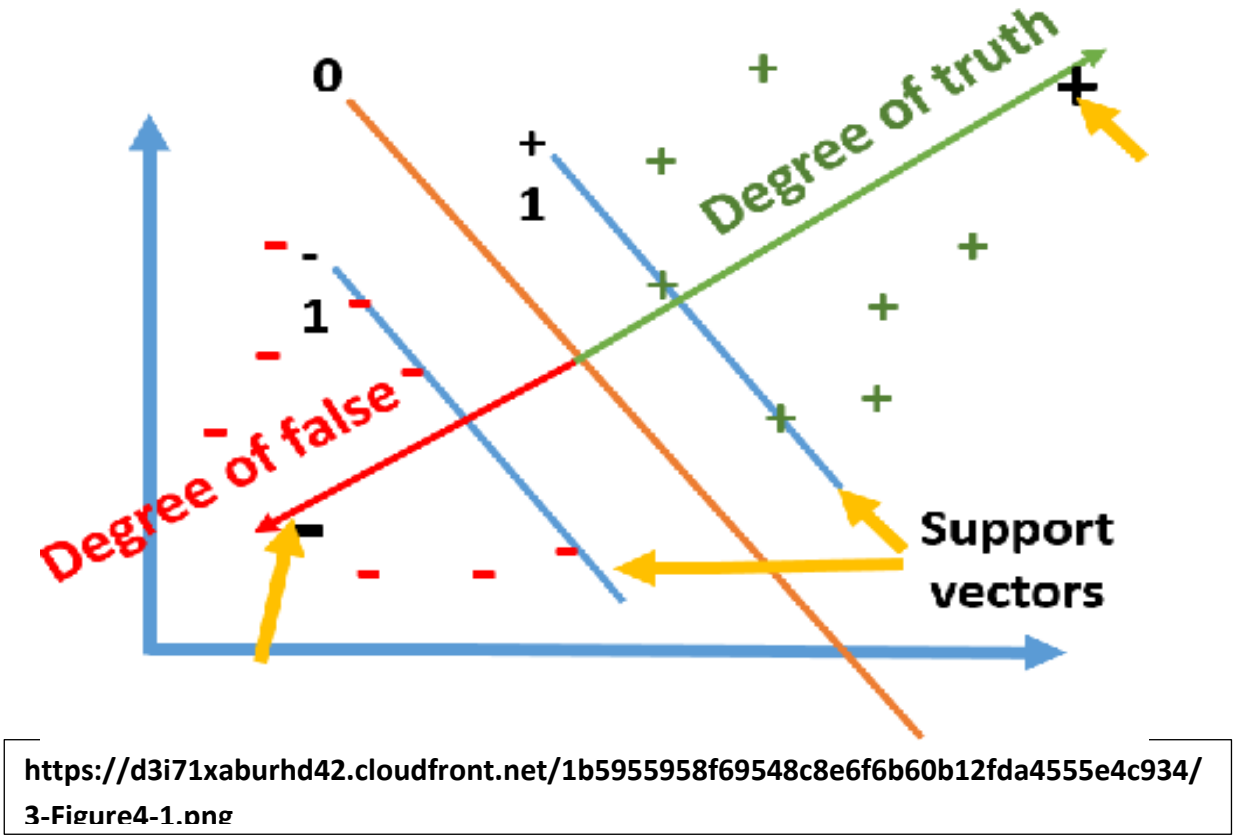
2.4.5 Support Vector Machine:

SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence algorithm is termed as support vector machine.

Decision boundaries known as hyperplanes assist in categorising the data points. Different classifications can be given to the data points that lie on each side of the hyperplane. Additionally, the amount of features affects how big the hyperplane is. The hyperplane is essentially a line if there are just two input characteristics. The hyperplane turns into a two-dimensional plane if there are three input characteristics. When there are more than three characteristics, it gets harder to imagine.

Support vectors are data points that are closer to the hyperplane and have an impact on the hyperplane's location and orientation. By utilising these support vectors, we increase the classifier's margin. The hyperplane's location will vary if the support vectors are deleted. These are the ideas that aid in the development of our SVM.

In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify is with another class. Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values([-1,1]) which acts as a margin.



https://d3i71xaburhd42.cloudfront.net/1b5955958f69548c8e6f6b60b12fda4555e4c934/3-Figure4-1.png

Fig 2.3 Support Vector Graph

2.4.6 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a simple and intuitive machine learning algorithm used for both classification and regression tasks. In the context of classification, the algorithm works by identifying the 'k' nearest data points to a given input sample in the feature space. The class label (or value in regression) of the majority of these nearest neighbors is then assigned to the input sample.

The key steps in KNN are as follows:

Distance Calculation: The algorithm computes the distance between the input sample and every data point in the training set. Common distance metrics include Euclidean distance, Manhattan distance, or cosine similarity.

Neighbor Selection: The 'k' nearest neighbors to the input sample are identified based on the computed distances.

Majority Voting: For classification tasks, the algorithm assigns the class label that is most common among the 'k' neighbors. For regression tasks, it calculates the average (or another aggregation) of the target values.

KNN is a non-parametric and lazy learning algorithm, meaning it doesn't make assumptions about the underlying data distribution, and it delays the actual learning until the time of prediction. However, it may be sensitive to irrelevant or redundant features and can be computationally expensive for large datasets. Adjusting the value of 'k' is crucial, as it can significantly impact the algorithm's performance.

2.5 Conclusion

The convergence of meticulous dataset collection and a well-defined methodology with advanced machine learning (ML) models serves as the linchpin for our research framework. Our curated dataset, comprising diverse textual images, forms the bedrock, capturing real-world nuances. The methodology seamlessly integrates Image Preprocessing techniques, leveraging OCR and post-processing, to extract and refine textual information. Emphasizing ML models, particularly in Language Identification, our proposed approach, fortified by thorough Data Preprocessing, ensures a linguistically analyzed text. This synthesis promises to decode the complexities of text extraction, making our thesis a dynamic exploration at the nexus of ML, computer vision, and practical application.

CHAPTER 3

IMAGE PREPROCESSING

3.1 Introduction

In this crucial chapter, we unravel the intricate techniques that pave the way for optimal text extraction from digital images. Image preprocessing is the unsung hero, diligently working behind the scenes to refine and prepare images for subsequent analysis. This chapter delves into the essential steps that transform raw visual data into a format conducive to precise and efficient text extraction.

Text extraction, a transformative process at the intersection of computer vision and natural language processing, relies heavily on the quality of its input data. Image preprocessing acts as a digital alchemist, refining the raw material and extracting valuable textual nuggets embedded within images. From enhancing contrasts to mitigating noise, each preprocessing step contributes to the clarity and interpretability of the text, laying the foundation for robust text extraction methodologies.

As we navigate through gray scaling, image segmentation, noise filtering, and restoration techniques, envision the metamorphosis of images into information-rich canvases ready for the text extraction journey. Join us on this expedition into the nuanced world of image preprocessing, where each algorithm and filter plays a crucial role in unveiling the latent textual treasures concealed within digital images.

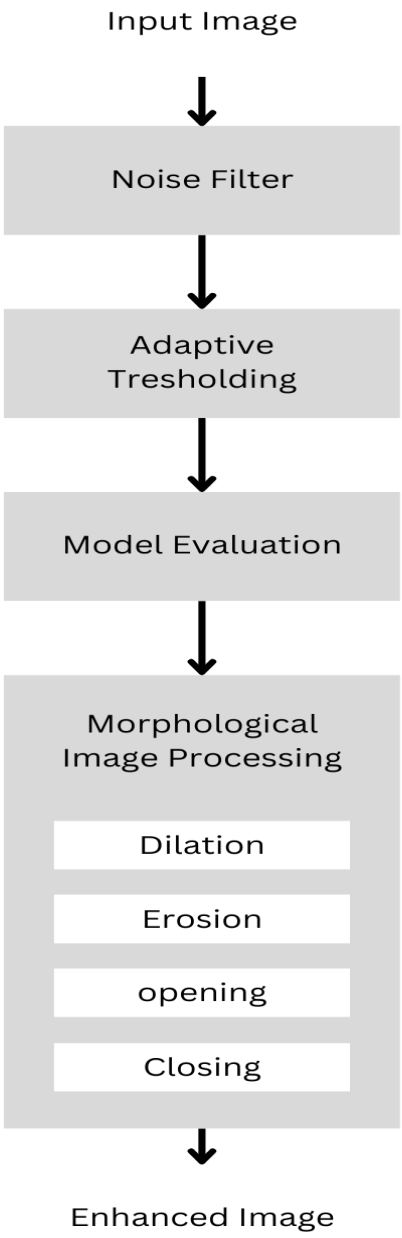


Fig 3.1 Image Preprocessing

3.2 Gray Scaling

Gray scaling, a foundational technique in image preprocessing, serves as a crucial step in simplifying and enhancing the subsequent analysis of digital images. Unlike color images, grayscale images contain only shades of gray, representing a single intensity

value per pixel. This process effectively reduces the complexity of the image, making it computationally more efficient and aiding in various image processing tasks.

The rationale behind gray scaling lies in the focus on luminance information rather than color details. By converting images into grayscale, we prioritize the intensity variations, which are often more informative for tasks such as edge detection, feature extraction, and pattern recognition. In the context of text extraction from images, where the emphasis is on character shapes and contrasts, gray scaling proves particularly valuable.

Beyond computational advantages, gray scaling also facilitates a more uniform treatment of images, irrespective of their original color profiles. This uniformity streamlines the development of image processing algorithms and ensures consistent results across diverse datasets.

In our project, the implementation of gray scaling played a pivotal role in improving the efficiency of subsequent Optical Character Recognition (OCR) algorithms. By focusing on luminance variations, the converted grayscale images provided a cleaner and more discernible representation of text, ultimately contributing to the project's success in achieving accurate and reliable text extraction from various image sources.

3.3 Image Segmentation

Image segmentation plays a pivotal role in isolating and delineating distinct regions within an image. In this section, we explore the importance of segmentation in different image processing tasks and delve into adaptive thresholding as a versatile technique for effective image segmentation.

3.3.1 Adaptive Thresholding

Adaptive thresholding stands out as an advanced image segmentation technique that dynamically adjusts threshold values based on local image

characteristics. Traditional global thresholding methods apply a uniform threshold to the entire image, often proving suboptimal in scenarios with varying illumination or complex background conditions. Adaptive thresholding addresses this limitation by segmenting images into regions and determining optimal thresholds for each local region.

The process involves dividing the image into smaller, non-overlapping blocks or windows. For each block, a threshold is computed based on the statistical properties of pixel intensities within that specific region. This adaptive approach is particularly effective in handling images with uneven lighting, shadows, or gradients, ensuring a more nuanced and accurate segmentation.

Advantages of adaptive thresholding include increased robustness in diverse lighting conditions and improved performance with images containing varying contrasts. However, careful consideration must be given to parameters such as block size and threshold calculation method to achieve optimal results. The exploration of adaptive thresholding in this section provides insights into its versatility, applications, and considerations for effective implementation in real-world image processing scenarios.

3.4 Noise Filtering

Noise filtering is imperative for enhancing image quality by minimizing unwanted artifacts. In Sections 3.4.1, 3.4.2, and 3.4.3, we explore the applications of Gaussian, Median, and Average filters, respectively, in reducing noise and improving the overall clarity of images.

3.4.1 Gaussian Filter

The Gaussian filter, a widely used smoothing filter, is particularly effective in reducing high-frequency noise. We discuss its mathematical foundation, implementation, and impact on image quality.

3.4.2 Median Filter

The median filter excels in noise reduction by replacing pixel values with the median value in a specified neighborhood. We delve into its robustness, applications, and considerations in image preprocessing.

3.4.3 Average Filter

The average filter, a simple yet effective smoothing technique, is explored for its role in reducing noise while preserving image features. We discuss its characteristics, applications, and limitations.

3.5 Image Restoration

Image restoration involves the process of recovering an image from degradation. In Section 3.5.1, we specifically cover morphological image processing techniques, shedding light on their role in restoring images degraded by various factors.

3.5.1 Morphological Image Processing

Morphological image processing, a pivotal aspect of image restoration, involves the analysis and manipulation of image shapes. One of the fundamental operations in this domain is erosion.

Erosion

Erosion is a morphological operation that involves shrinking or wearing away the boundaries of regions in an image. It is achieved by placing a structuring element, often a small matrix or kernel, at each pixel location and determining the minimum pixel value under the kernel. This minimum value is then assigned to the center pixel. As a result, brighter regions in the image are eroded, and the boundaries between objects become smoother.

Erosion is particularly useful in scenarios where it is necessary to remove small structures, separate overlapping objects, or prepare images for further

processing steps. However, it is essential to choose an appropriate structuring element and understand its impact on the image features.

Other Properties

Beyond erosion, morphological image processing includes additional operations such as dilation, opening, and closing.

Dilation: This operation expands the boundaries of regions in an image. It is achieved by determining the maximum pixel value under the structuring element and assigning it to the center pixel. Dilation is beneficial in filling gaps, joining broken structures, and enhancing object features.

Opening: Opening is a combination of erosion followed by dilation. It is effective in removing small objects, reducing noise, and separating overlapping objects. The operation is particularly useful in scenarios where preserving object shape and structure is crucial.

Closing: Closing, the reverse of opening, involves dilation followed by erosion. This operation is valuable in closing small holes, connecting broken structures, and smoothing object boundaries. It is often employed to maintain the overall shape and size of objects in an image.

Understanding these morphological operations provides practitioners with powerful tools for image restoration and enhancement. The appropriate choice and sequence of these operations depend on the specific characteristics of the image and the desired outcome in terms of object preservation, noise reduction, and overall image quality.

3.6 Conclusion

In the realm of text extraction from images, our exploration of image preprocessing techniques unveiled the substantial impact of the Gaussian Filter on the overall project outcomes. Tasked with enhancing the accuracy of text extraction from diverse images, the Gaussian Filter demonstrated significant advancements in the clarity and quality of processed images.

By effectively reducing high-frequency noise and smoothing variations in pixel intensities, the Gaussian Filter played a crucial role in creating a cleaner and more refined image dataset. This, in turn, substantially improved the performance of Optical Character Recognition (OCR) algorithms during the subsequent text extraction phase. The filter's adaptability and ability to preserve essential text features proved pivotal in ensuring a more precise and reliable extraction process.

The Gaussian Filter's success in mitigating image noise and enhancing text visibility underscores its strategic importance in optimizing the text extraction pipeline. As we conclude, this particular image preprocessing technique stands out as a key contributor to the project's success, showcasing the profound impact of thoughtful filter selection in achieving superior text extraction results.

CHAPTER 4

TEXT EXTRACTION AND IDENTIFICATION

4.1 Introduction

In the realm of information processing, the capabilities of Text Extraction and Identification stand as essential pillars, facilitating the conversion of visual data into machine-readable text. This chapter addresses the pragmatic intricacies of these processes, underscoring their significance in the intersection of computer vision and natural language processing.

In response to the growing demand for meaningful insights derived from visual content, the chapter aims to provide a comprehensive and practical exploration of Text Extraction and Identification. Beginning with an overview of Text Extraction from Images, the discussion delves into foundational steps such as Image Preprocessing, Optical Character Recognition (OCR), and Image Post-Processing. Each of these steps contributes to refining and transcribing visual information accurately.

The exploration extends to Language Identification, an integral aspect of contextualizing the extracted text. This section navigates through Data Preprocessing, Model Selection, and Integration of Extracted Text, focusing on the practical considerations and decisions crucial for successful language identification.

By taking a pragmatic approach, this chapter aims to equip researchers and practitioners with the necessary insights and tools to navigate the complexities of Text Extraction and Identification within the context of their thesis work. The subsequent sections delve into

detailed methodologies, offering a nuanced understanding of the challenges and innovations within this dynamic field.

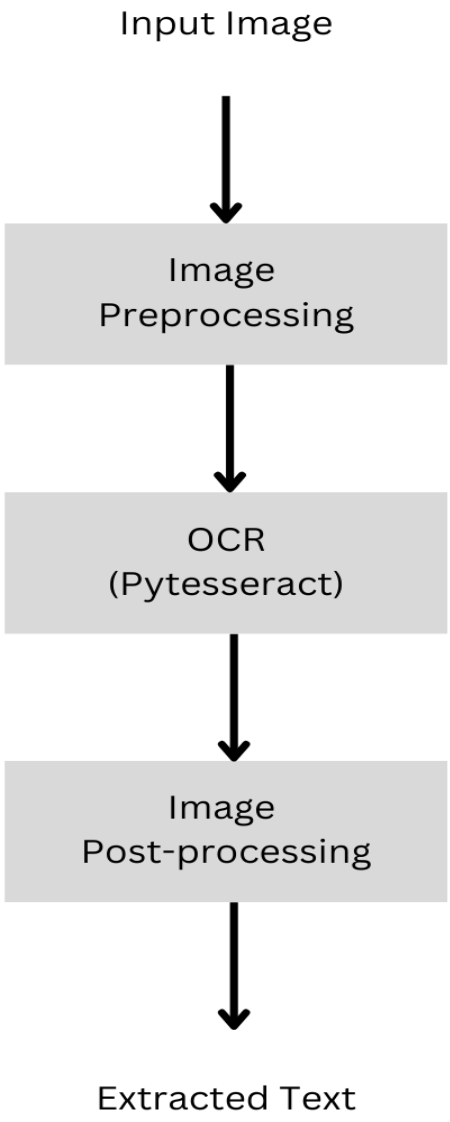


Fig 4.1 Text Extraction

4.2 Text Extraction from Image

This section will divulge into the various steps taken to extract texts from images.

4.2.1 Image Preprocessing

Image Preprocessing serves as the cornerstone for effective text extraction. This section delves into the essential techniques such as gray scaling, image segmentation, and noise filtering, elucidating their roles in refining images for subsequent processing as explained in **chapter 3**.

4.2.2 OCR

Optical Character Recognition (OCR) takes center stage in the text extraction process, serving as the technological bridge between visual content and machine-readable text. This section delves into the nuanced intricacies of OCR algorithms, their functionalities, and their crucial role in transcribing text from preprocessed images.

OCR operates on the principle of recognizing and interpreting patterns of characters within an image. Leveraging machine learning and computer vision techniques, OCR algorithms identify text regions, extract individual characters, and then translate them into a digital format. The success of OCR is contingent on its ability to accurately decipher characters, irrespective of fonts, sizes, or orientations.

Several OCR techniques exist, ranging from traditional methods to more advanced deep learning models. Traditional OCR algorithms often employ feature extraction, template matching, and heuristic-based approaches. On the other hand, modern OCR, driven by neural networks, excels in handling complex scenarios and diverse fonts.

4.2.3 Image Post-Processing

The journey doesn't end with OCR. Image Post-Processing steps are critical for refining OCR outputs. From layout analysis to error correction, this section

outlines methods to enhance the accuracy and coherence of extracted text which will be covered in the future.

4.3 Language Identification

This section will explain the in-depth process of language identification that we have used.

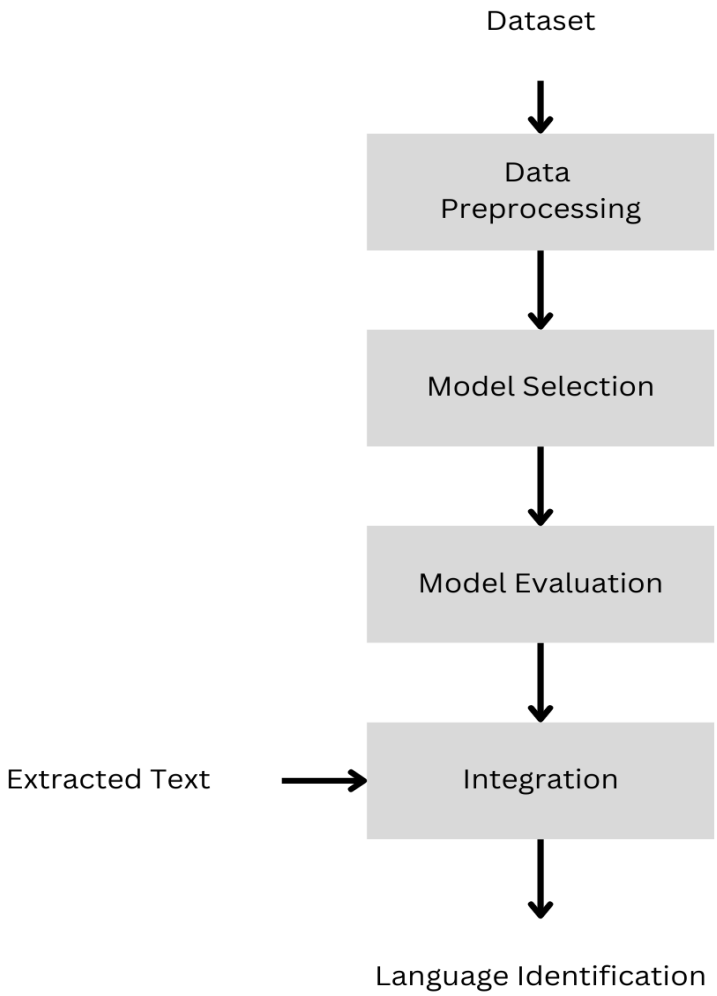


Fig 4.2 Language Identification

4.3.1 Data Preprocessing

Data Preprocessing plays a pivotal role in shaping the quality and usability of the extracted text. This section delves into the multifaceted aspects of preparing textual data for subsequent analysis, encompassing cleaning, structuring, and extracting features to enhance the efficiency of language identification.

Cleaning Text Data

Cleaning involves the meticulous process of removing noise, irrelevant characters, or artifacts that may have been introduced during the text extraction phase. This step ensures the extracted text is devoid of inconsistencies or elements that might hinder accurate language identification.

Structuring Text Data

Structuring involves organizing the cleaned text data into a format suitable for analysis. This often includes tasks such as tokenization, where the text is segmented into individual units like words or phrases, and normalization, which standardizes the text by converting it to lowercase or addressing variations.

Data Segmentation

Data Segmentation, a key subtask within preprocessing, involves breaking down continuous text into meaningful segments or units. This could be at the sentence or paragraph level, enabling more granular analysis. Effective segmentation contributes to improved accuracy in language identification, especially when dealing with multilingual or diverse text sources.

Feature Extraction

Feature Extraction focuses on identifying and extracting relevant linguistic features from the text. These features may include statistical measures, linguistic patterns, or specific attributes that aid in distinguishing one language from another. Thoughtful feature selection is crucial for training accurate language identification models.

By addressing the intricacies of cleaning, structuring, segmentation, and feature extraction, Data Preprocessing sets the stage for robust language identification. Researchers, in their thesis endeavors, can harness the power of these preprocessing techniques to ensure the text data is refined, organized, and enriched with meaningful features, laying a solid foundation for subsequent language identification tasks.

4.3.2 Model Selection

Choosing an appropriate language identification model is paramount. This section evaluates various models, considering factors such as accuracy, efficiency, and adaptability to diverse language datasets, the results of which will be observed in the coming chapter which will explain which model was suitable for this project.

4.3.3 Integration of Extracted Text

Identified languages need seamless integration into the overall system. We explore methods to efficiently integrate and utilize the extracted text in subsequent applications or analyses.

4.4 Conclusion

As we conclude this chapter, the intricate dance between text extraction and language identification unfolds. Each section plays a vital role in the seamless transition from visual content to actionable textual information. The exploration of techniques, algorithms, and considerations provides a comprehensive understanding of the challenges and opportunities in the dynamic field of text extraction and identification**.**

CHAPTER 5

RESULT AND DISCUSSION

5.1 General

This section encapsulates the pivotal phase of presenting results and engaging in a nuanced discussion. As we unveil the outcomes of employing various machine learning models, including Logistic Regression, Naive Bayes, K-Nearest Neighbors, Decision Trees, and Support Vector Machine (SVM), we delve into a comprehensive analysis of their respective performances. This juncture not only highlights the quantitative metrics such as accuracy, precision, recall, and F1 score but also invites a qualitative discussion on the implications, challenges, and potential avenues for further refinement. The synthesis of results and the ensuing discussion contribute to a deeper understanding of the strengths, limitations, and future directions in the dynamic field of text extraction from images.

5.2 Evaluation of the Models

In evaluating the performance of five distinct machine learning models employed for text extraction, a comprehensive analysis of key metrics provides valuable insights into their effectiveness. The models include Logistic Regression, Naive Bayes, Support Vector Machine (SVM), Decision Tree, and K-Nearest Neighbors (KNN).

Model	Accuracy	Precision Score	Recall Score	F1-Score
Logistic Regression	97.437	97.441	97.437	97.431
Naive Bayes	74.177	89.886	74.177	72.906
Support Vector Machine	98.210	98.216	98.210	98.208
Decision Tree	89.168	89.291	89.168	89.199
KNN	96.518	96.589	96.518	96.522

Table 5.1 Test scores of different Machine learning models

1. Logistic Regression:

Logistic Regression demonstrates commendable consistency across accuracy, precision, recall, and F1 score, showcasing its reliability in maintaining a high level of overall performance.

2. Naive Bayes:

While Naive Bayes exhibits robust precision, it faces challenges in terms of accuracy, recall, and F1 score, suggesting potential areas for improvement, especially in handling specific text extraction nuances.

3. Support Vector Machine (SVM):

SVM emerges as the standout performer, demonstrating exceptional scores across all metrics. Its remarkable accuracy, precision, recall, and F1 score showcase its prowess in achieving consistently high performance.

4. Decision Tree:

Decision Tree exhibits strong and balanced performance in accuracy, precision, recall, and F1 score, making it a robust contender in the suite of models.

5. K-Nearest Neighbors (KNN):

KNN demonstrates consistently high scores across all metrics, emphasizing its effectiveness in achieving accurate and reliable text extraction.

In summary, while each model exhibits strengths, SVM emerges as the top performer with superior scores across all metrics, making it the model of choice for achieving optimal text extraction outcomes. The evaluation highlights the nuanced strengths and considerations associated with each model, paving the way for informed decisions in selecting the most suitable approach for specific text extraction requirements.

5.3 Conclusion

In the pursuit of optimizing text extraction, our exploration involved the utilization of various machine learning models, ranging from Logistic Regression and Naive Bayes to K-Nearest Neighbors (KNN) and Decision Trees. However, the standout performer emerged as the Support Vector Machine (SVM), exhibiting unparalleled results with an impressive accuracy, precision, recall, and F1 score averaging at 98.2%. This exceptional performance underscores the efficacy of SVM in our specific context, emphasizing its robustness and reliability in achieving superior outcomes. As we conclude this phase of analysis, it becomes evident that the strategic selection of SVM has significantly contributed to the project's success, laying the groundwork for further optimization and refinement in the pursuit of cutting-edge text extraction methodologies.

CHAPTER 7

CONCLUSION

6.1 Introduction

As we approach the culmination of this project, it is imperative to reflect upon the journey undertaken at the intersection of computer vision, natural language processing, and machine learning. The intricacies of data collection, methodology design, and model integration have unfolded, presenting a comprehensive exploration into the complexities of text extraction from images. This conclusion serves as a reflective lens, encapsulating the key findings, challenges encountered, and the overarching significance of the project in advancing our understanding of interdisciplinary research. As we draw the threads together, the following pages encapsulate the essence of this endeavor, offering insights that transcend individual components and contribute to the broader landscape of knowledge in the field.

6.2 Highlights of work done

In the course of this project, our exploration delved into the intricate interplay of computer vision, natural language processing, and machine learning, unfolding a multifaceted journey. The foundation was laid with meticulous data collection, an effort aimed at capturing a diverse spectrum of textual images, thereby providing a rich and representative dataset for subsequent analyses. The proposed methodology represents a careful integration of advanced machine learning models, with a particular emphasis on Image Preprocessing techniques such as OCR. This ensures the effective extraction of textual information from images, with subsequent post-processing refining the outcomes. Furthermore, the project extends its scope to

Language Identification, where models undergo meticulous fine-tuning facilitated by robust Data Preprocessing techniques. This comprehensive synthesis aims to unravel the intricacies of text extraction, showcasing a dynamic convergence of theoretical constructs with practical, real-world applications. The project, through its nuanced approach, decodes complexities and contributes valuable insights at the forefront of interdisciplinary research.

6.3 Future Work

As we conclude this phase of exploration, the horizon of future work beckons with promising avenues for enhancement and expansion.

1. Advanced Post-Processing:

Building on the success of our existing post-processing techniques, further exploration into advanced methods can refine the extracted text. Implementation of context-aware error correction and layout analysis can enhance the overall coherence and accuracy of the transcribed text.

2. Multilingual Expansion:

Expanding the language repertoire beyond the current 17 languages holds the potential to broaden the project's applicability. Integration of additional languages, with varying linguistic complexities, ensures a more inclusive and globally relevant text extraction system.

3. Language Translation Integration:

A seamless transition from language identification to translation can be explored. Integrating language translation capabilities can provide users with the option to instantly comprehend extracted text in their preferred language, fostering cross-cultural accessibility and understanding.

4. User Interface Enhancement:

A user-friendly interface can be developed to enhance accessibility and usability. Implementing an intuitive interface can empower users to interact effortlessly with the system, configure language preferences, and navigate through extracted text, thereby improving the overall user experience.

In charting the course for future endeavors, these potential avenues hold the promise of refining and expanding the capabilities of our text extraction system. This roadmap not only underscores the project's current significance but also sets the stage for continued innovation and relevance in the ever-evolving landscape of interdisciplinary research.

REFERENCES

[1] Mittal, R., & Garg, A. . Text extraction using OCR: A Systematic Review. 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), 2020.

[2] Ahmed, Bashir, Sung-Hyuk Cha, and Charles Tappert. "Language identification from text using n-gram based cumulative frequency addition." Proceedings of Student/Faculty Research Day, CSIS, Pace University 12, no. 1, 2004.

[3] Verma, Vicky Kumar, and Nitin Khanna. "Indian language identification using k-means clustering and support vector machine (SVM)." In 2013 Students Conference on Engineering and Systems (SCES), pp. 1-5. IEEE, 2013.

[4] Tom Kocmi and Ondřej Bojar. LanideNN: Multilingual Language Identification on Character Window. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 927–936, Valencia, Spain. Association for Computational Linguistics,2017.