

TEXT EXTRACTION

MALLA REDDY UNIVERSITY

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

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (AI & ML)

Submitted by

D. Sri Lasya : 2211CS020112

D. Trishul Raj : 2211CS020114

D. Srimanth Reddy : 2211CS020115

D. Mourya Sri : 2211CS020116

D. Shivaraj : 2211CS020117

Under the Guidance of

Prof. Siva Kumar

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)



2024



MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

COLLEGE CERTIFICATE

This is to certify that this is the bonafide record of the application development entitled, “**Text Extraction**” submitted by D. Sri Lasya (2211CS020112), D. Trishul Raj (2211CS020114), D. Srimanth Reddy (2211CS020115), D. Mourya Sri (2211CS020116), D. Shivaraj (2211CS020117) B. Tech III year I semester, Department of CSE (AI &ML) during the year 2024-2025. The results embodied in the report havenot been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE

HEAD OF THE DEPARTMENT

DEAN CSE(AI&ML)
Dr. Thayyaba Khatoon

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all the efforts with success. We are very grateful to our project mentor Prof A. Siva Kumar, for the guidance, inspiration and constructive suggestions that helped us in the development of this application.

We are much obliged to Prof. Siva Kumar (Application Development Incharge) for encouraging and supporting us immensely by giving many useful inputs with respect to the topic chosen by us, throughout the development of the application ensuring that our project is a success.

We also express our heartfelt gratitude to Dr. Thayyaba Khatoon (Dean (AIML)), for giving all of us such a wonderful opportunity to explore ourselves and the outside world to work on the real-life scenarios where the machine learning is being used nowadays.

We also thank our parents and family at large for their moral and financial support in funding the project to ensure successful completion of the project.

ABSTRACT

This project focuses on text extraction, a critical task that converts unstructured data into structured information using advanced deep learning algorithms. As digital content continues to grow, efficient text extraction from sources such as documents, web pages, and images becomes increasingly important. The project aims to develop a robust system leveraging deep learning frameworks, including TensorFlow and Keras, to enhance the speed and accuracy of text extraction. We will address challenges in the extraction process, such as language variations, format differences, and data noise. Our approach includes a multi-step pipeline featuring preprocessing, feature extraction, and classification using deep learning models. We will also prioritize semantic accuracy to preserve the meaning and context of extracted information. A user-friendly interface will be developed to allow for customization of extraction parameters and effective visualization of results. The system will incorporate advanced analytics for deeper insights into the data. The project's outcomes are intended to improve text extraction efficiency and contribute to data mining and information retrieval. This will support applications in academic research, business intelligence, and content management, ultimately enhancing decision-making processes across various fields.

CONTENTS

CHAPTER NO.	TITLE	PAGE
1.	INTRODUCTION:	
	1.1 Project Identification / Problem Definition	1
	1.2 Objective of project	1
	1.3 Scope of the project	2
2.	ANALYSIS:	
	2.1 Project Planning and Research	3
	2.2 Software requirement specification	3
	2.2.1 Software requirement	3-4
	2.2.2 Hardware requirement	5
	2.3 Model Selection and Architecture	6-8
3.	DESIGN:	
	3.1 Introduction	9
	3.2 DFD/ER/UML diagram (any other project diagram)	9
	3.3 Data Set Descriptions	10
	3.4 Data Preprocessing Techniques	10
	3.5 Methods & Algorithms	11
4.	DEPLOYMENT AND RESULTS:	
	4.1 Introduction	12
	4.2 Source Code	12-14
	4.3 Model Implementation and Training	14
	4.4 Model Evaluation Metrics	15
	4.5 Model Deployment: Testing and Validation	15
	4.6 Web GUI's Development	16-18
	4.7 Results	18-21
5.	CONCLUSION:	
	5.1 Project conclusion	22
	5.2 Future Scope	22

1 INTRODUCTION

1.1 PROBLEM DEFINITION:

The Accurate text extraction from images is challenging due to variations in fonts, backgrounds, and image quality. A robust solution is needed to handle diverse conditions and improve extraction accuracy. The goal is to develop a system that seamlessly integrates into existing workflows with minimal manual intervention.

1.2 OBJECTIVE OF THE PROJECT:

The objective of a **Text Extraction** project is to develop an automated system capable of identifying, extracting, and retrieving relevant information from unstructured or semi-structured textual data sources. This system aims to transform raw text data into a structured format, making it accessible for analysis, decision-making, and data processing. By automating information extraction, the project seeks to reduce manual effort, improve efficiency, and allow users to retrieve specific information quickly from large volumes of text, such as names, dates, or product details. The extracted data is organized into structured formats like JSON or CSV, facilitating seamless integration into data pipelines and enabling advanced analytics and machine learning applications, including sentiment analysis, topic modeling, and predictive modeling. Ultimately, this project enhances decision-making by providing accurate, contextually relevant information and is designed with scalability in mind, allowing it to accommodate diverse data sources, languages, and document types across different applications and use cases. Through these objectives, the text extraction project supports increased productivity, data-driven decisions, and valuable insights from text-based data sources.

1.3 SCOPE AND LIMITATIONS OF THE PROJECT:

Scope:

1. Automates the identification and extraction of structured information from unstructured or semi-structured text data.
2. Covers various data sources, including PDFs, Word documents, web pages, emails, and scanned images.
3. Supports extraction of specific information like dates, names, product details, and numeric values.
4. Utilizes techniques like Natural Language Processing (NLP) and Optical Character Recognition (OCR) for effective data processing.
5. Employs rule-based or machine learning approaches to manage diverse document formats and complexities.
6. Transforms extracted information into structured formats (e.g., JSON, CSV, databases) for analysis, reporting, and data integration.

Limitations:

1. Accuracy may be affected by poor document quality, such as blurry images or scanned text with noise.
2. Limited ability to interpret complex sentence structures or context-dependent meanings.
3. Performance may vary across languages, especially if trained on specific languages only.
4. Struggles with handwritten or stylized fonts, which may not be accurately captured by OCR.
5. Sensitive to variations in document formatting, such as different layouts or custom templates.
6. Computationally intensive, which can lead to slower processing times with large datasets.
7. Requires continuous updates and retraining for high performance, particularly if new document types or languages are introduced.

2.ANALYSIS

2.1 PROJECT PLANNING AND RESEARCH

The **Project Planning and Research** phase of a Text Extraction project is foundational, setting the stage for effective development and deployment. This phase begins by clearly defining the project's goals, such as the types of information to be extracted, the targeted data sources, and the system's purpose—whether for analytics, automation, or decision support. Next, a thorough requirement analysis is conducted by engaging with stakeholders to understand document types, required accuracy, and specific entities like dates, names, or values. Selecting appropriate tools and technologies is another critical step, where NLP frameworks, OCR tools, and programming languages are evaluated based on compatibility, scalability, and integration needs. Additionally, market and technology research is performed to assess existing solutions and industry standards, helping to identify any potential limitations and align the project with best practices and technological advancements. Together, these planning and research activities provide a structured roadmap that guides the project's design and development.

2.2 SOFTWARE REQUIREMENT SPECIFICATION:

2.2.1 SOFTWARE REQUIREMENT:

Programming Languages: Python, Java, or other languages with strong support for NLP and OCR libraries.

Natural Language Processing (NLP) Libraries: NLTK, spaCy, or Hugging Face Transformers for text processing and entity recognition.

Optical Character Recognition (OCR) Tools: Tesseract OCR or Google Cloud Vision API for extracting text from images or scanned documents.

Machine Learning Frameworks: TensorFlow, PyTorch, or Scikit-learn for building and training models if custom extraction models are needed.

Data Storage: SQL databases (e.g., MySQL, PostgreSQL) or NoSQL options (e.g., MongoDB) for structured storage of extracted data.

Web Scraping Tools (if needed): BeautifulSoup, Scrapy, or Selenium for extracting text data from web pages.

Data Preprocessing Libraries: Pandas and NumPy for cleaning, organizing, and transforming extracted data.

API Integration Tools: Tools like Flask or FastAPI for creating APIs to integrate the extraction system with other applications.

Version Control System: Git for managing code versions and collaboration among development team members.

Integrated Development Environment (IDE): Visual Studio Code, PyCharm, or Jupyter Notebook for efficient coding and testing.

Cloud Services (Optional): AWS, Google Cloud, or Azure for scalability, model hosting, and processing large datasets if needed.

Error Logging and Monitoring: Tools like Log4j, ELK Stack, or Sentry to monitor performance and troubleshoot issues in real time.

2.2.2 HARDWARE REQUIREMENT:

Hardware Requirements for a Text Extraction Project:

1. **Processor:** Multi-core processor (Intel i5/i7 or AMD Ryzen 5/7 or higher) to handle data processing and model training.
2. **Memory (RAM):** Minimum of 16 GB RAM for efficient handling of large datasets; 32 GB or more recommended for complex models.
3. **Storage:** At least 500 GB SSD for faster data read/write speeds; higher capacity may be needed depending on data volume.
4. **Graphics Processing Unit (GPU):** Dedicated GPU (NVIDIA RTX 3060 or higher) if using deep learning models, as it accelerates training and inference.
5. **Network Connectivity:** High-speed internet connection for accessing cloud services, APIs, and external data sources.
6. **Backup Storage (Optional):** External HDD/SSD or cloud-based storage for data backup and archiving of processed documents.
7. **Power Supply and Cooling:** Reliable power supply and proper cooling mechanisms, especially if using high-performance GPUs for extended periods.
8. **Cloud Hardware (Optional):** Access to cloud computing resources (AWS, Google Cloud, Azure) for additional processing power, especially for large-scale deployments.

2.3 MODEL SELECTION AND ARCHITECTURE

MODEL SELECTION

1. Convolutional Neural Networks (CNNs):

CNNs are highly effective for image analysis tasks. They can automatically extract and learn features from images containing text, enabling accurate text recognition.

- Examples: Architectures like VGGNet, ResNet, and Inception are widely used for Optical Character Recognition (OCR) tasks.

2. Transfer Learning:

Utilize pre-trained CNN models (e.g., Tesseract, VGG16) fine-tuned on text-specific datasets to leverage existing feature extraction capabilities, enhancing accuracy, especially with smaller datasets.

3. Support Vector Machines (SVMs):

When used with features extracted from images (e.g., Histogram of Oriented Gradients), SVMs can classify text regions by finding the optimal hyperplane that separates different classes of text.

4. Ensemble Methods:

Combine predictions from models trained on different aspects of text recognition (e.g., CNNs for feature extraction, RNNs for sequence modeling) to improve overall accuracy in text extraction.

5. Multi-Modal Neural Networks:

Integrate features from various data sources (e.g., images with contextual information) into a unified model to enhance text recognition. This can involve multi-stream CNNs or attention-based architectures that consider both visual and contextual data.

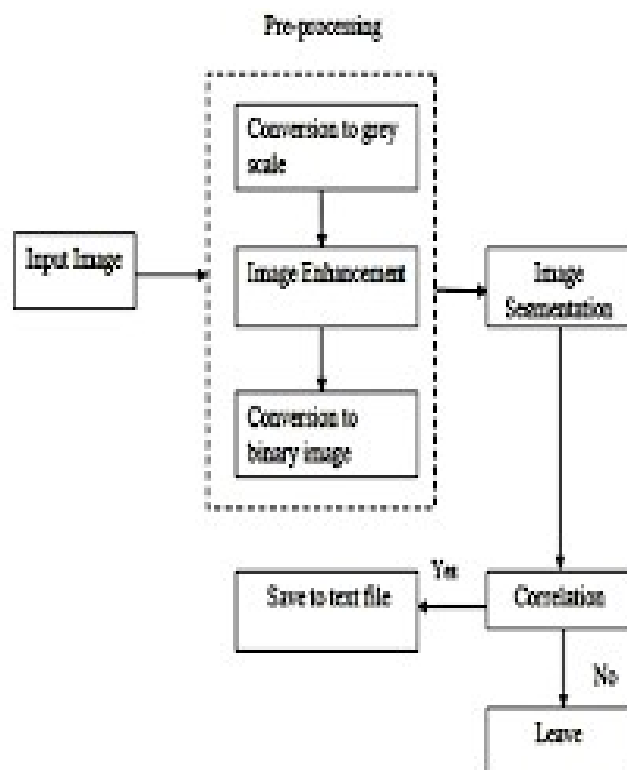
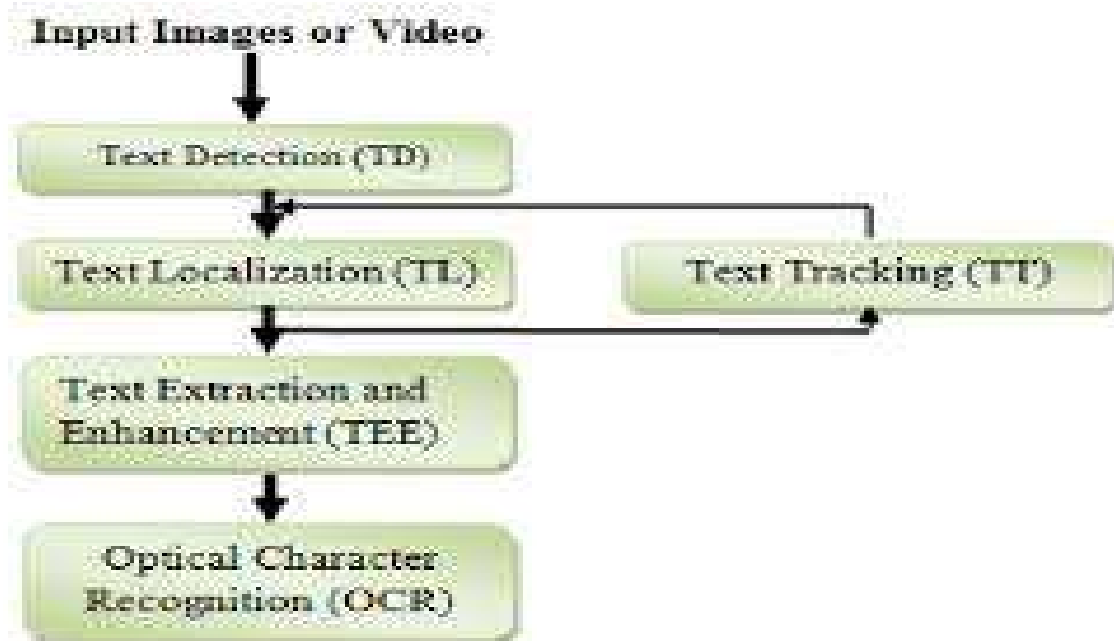
6. Attention Mechanisms:

Implement attention layers to focus on crucial features in the image, enhancing the model's ability to prioritize relevant information for text extraction and improve recognition accuracy.

7. Transformer Models:

Although primarily used in NLP, transformers can be adapted for image text recognition by effectively handling complex relationships between different features, facilitating more nuanced

ARCHITECTURE:

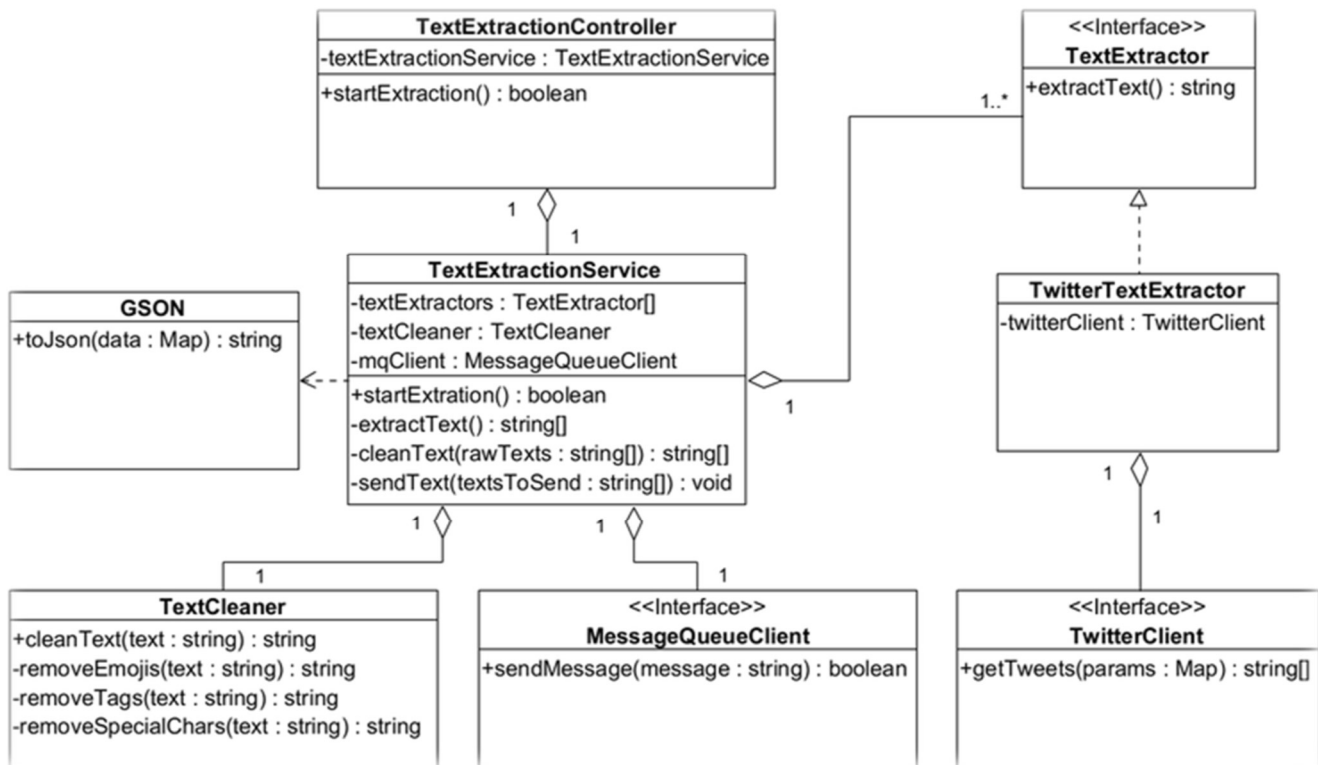


3.DESIGN

3.1 INTRODUCTION

The design phase of a Text Extraction project focuses on creating an architecture that efficiently identifies and retrieves specific information from unstructured or semi-structured text. This phase involves selecting the appropriate frameworks, setting up data processing pipelines, and defining workflows for document ingestion, text recognition, and data structuring. A well-planned design ensures scalability, accuracy, and smooth integration with other systems, forming the foundation for effective text extraction and data processing.

3.2 DFD/ER/UML DIAGRAM



3.3 DATA SET DESCRIPTIONS

Source Types: Various data formats, including PDFs, Word documents, web pages, emails, and images.

Data Fields: Information to be extracted, such as names, dates, locations, product details, and numeric values.

Data Structure: Unstructured or semi-structured data with varying formats, layouts, and styles.

Document Complexity: Ranges from simple text-based documents to complex, multi-page files with tables, images, or mixed content.

Language and Character Variability: Documents may be in multiple languages or contain special characters, requiring multilingual support.

3.4 DATA PREPROCESSING TECHNIQUES

Optical Character Recognition (OCR): Used for converting text from images or scanned documents into machine-readable text.

Text Preprocessing: Steps like tokenization, stemming, and lemmatization to clean and prepare raw text data.

Named Entity Recognition (NER): Identifies entities like names, dates, and locations within text.

Text Segmentation: Divides text into manageable sections (e.g., paragraphs, sentences) for easier processing.

Document Layout Analysis: Analyzes structure and layout to locate specific sections, tables, or fields within documents.

Data Normalization: Standardizes extracted data into consistent formats, such as date formats or numerical values.

Filtering and Deduplication: Removes irrelevant or duplicate information to enhance extraction accuracy.

3.5 METHODS AND ALGORITHMS

Rule-Based Extraction: Uses predefined rules (e.g., regular expressions) to locate specific patterns or keywords in text.

Machine Learning Models: Trains models to identify and extract entities based on labeled examples, improving with more data.

Deep Learning Techniques: Techniques like LSTM, BERT, or Transformers for complex language understanding and text extraction.

Hybrid Approach: Combines rule-based and machine learning methods for improved accuracy and adaptability.

Clustering and Classification: Groups similar documents or text segments, useful for categorizing and prioritizing data.

Sequence Labeling Algorithms: Algorithms like Conditional Random Fields (CRF) or Bi-LSTM CRF for tagging and extracting sequential data.

Knowledge Graphs: Organizes extracted entities and their relationships for enhanced data context and interconnections.

4.DEPLOYMENT AND RESULTS

4.1INTRODUCTION

The deployment phase of a Text Extraction project involves implementing the designed system into a production environment where it can operate effectively in real-world scenarios. This process includes setting up the necessary infrastructure, integrating the extraction system with existing data workflows, and ensuring that all components function seamlessly together. Rigorous testing is conducted to validate the accuracy and performance of the extraction algorithms, followed by user training to maximize system utilization. Once deployed, the system begins processing documents, and results are monitored to assess extraction quality and identify areas for improvement. Overall, this phase ensures that the text extraction solution delivers reliable outputs and meets the initial project objectives while allowing for ongoing adjustments based on user feedback and operational demands.

4.2 SOURCE CODE

```
import pandas as pd
import numpy as np

from glob import glob
from tqdm.notebook import tqdm

import matplotlib.pyplot as plt
from PIL import Image

plt.style.use('ggplot')

annot = pd.read_parquet('../input/textocr-text-extraction-from-images-dataset/annot.parquet')
imgs = pd.read_parquet('../input/textocr-text-extraction-from-images-dataset/img.parquet')
img_fns = glob('../input/textocr-text-extraction-from-images-dataset/train_val_images/train_images/*')

fig, ax = plt.subplots(figsize=(10, 10))
ax.imshow(plt.imread(img_fns[0]))
ax.axis('off')
plt.show()

image_id = img_fns[0].split('/')[-1].split('.')[0]
```

```

annot.query('image_id == @image_id')

fig, axs = plt.subplots(5, 5, figsize=(20, 20))
axs = axs.flatten()
for i in range(25):
    axs[i].imshow(plt.imread(img_fns[i]))
    axs[i].axis('off')
    image_id = img_fns[i].split('/')[-1].rstrip('.jpg')
    n_annot = len(annot.query('image_id == @image_id'))
    axs[i].set_title(f'{image_id} - {n_annot}')
plt.show()

import pytesseract

# Example call
print(pytesseract.image_to_string(img_fns[11], lang='eng'))

fig, ax = plt.subplots(figsize=(10,10))
ax.imshow(plt.imread(img_fns[11]))
ax.axis('off')
plt.show()

import easyocr

reader = easyocr.Reader(['en'], gpu = True)
results = reader.readtext(img_fns[11])
!pip install keras-ocr -q
import keras_ocr

pipeline = keras_ocr.pipeline.Pipeline()
results = pipeline.recognize([img_fns[11]])
pd.DataFrame(results[0], columns=['text', 'bbox'])
# easyocr
reader = easyocr.Reader(['en'], gpu = True)

dfs = []
for img in tqdm(img_fns[:25]):
    result = reader.readtext(img)
    img_id = img.split('/')[-1].split('.')[0]
    img_df = pd.DataFrame(result, columns=['bbox', 'text', 'conf'])
    img_df['img_id'] = img_id
    dfs.append(img_df)
easyocr_df = pd.concat(dfs)

# keras_ocr
pipeline = keras_ocr.pipeline.Pipeline()

dfs = []

```

```

for img in tqdm(img_fns[:25]):
    results = pipeline.recognize([img])
    result = results[0]
    img_id = img.split('/')[-1].split('.')[0]
    img_df = pd.DataFrame(result, columns=['text', 'bbox'])
    img_df['img_id'] = img_id
    dfs.append(img_df)
kerasocr_df = pd.concat(dfs)

def plot_compare(img_fn, easyocr_df, kerasocr_df):
    img_id = img_fn.split('/')[-1].split('.')[0]
    fig, axs = plt.subplots(1, 2, figsize=(15, 10))

    easy_results = easyocr_df.query('img_id == @img_id')[['text', 'bbox']].values.tolist()
    easy_results = [(x[0], np.array(x[1])) for x in easy_results]
    keras_ocr.tools.drawAnnotations(plt.imread(img_fn),
                                   easy_results, ax=axs[0])
    axs[0].set_title('easyocr results', fontsize=24)

    keras_results = kerasocr_df.query('img_id == @img_id')[['text', 'bbox']].values.tolist()
    keras_results = [(x[0], np.array(x[1])) for x in keras_results]
    keras_ocr.tools.drawAnnotations(plt.imread(img_fn),
                                   keras_results, ax=axs[1])
    axs[1].set_title('keras_ocr results', fontsize=24)
    plt.show()
# Loop over results
for img_fn in img_fns[:25]:
    plot_compare(img_fn, easyocr_df, kerasocr_df)

```

4.3 MODEL IMPLEMENTATION AND TRAINING

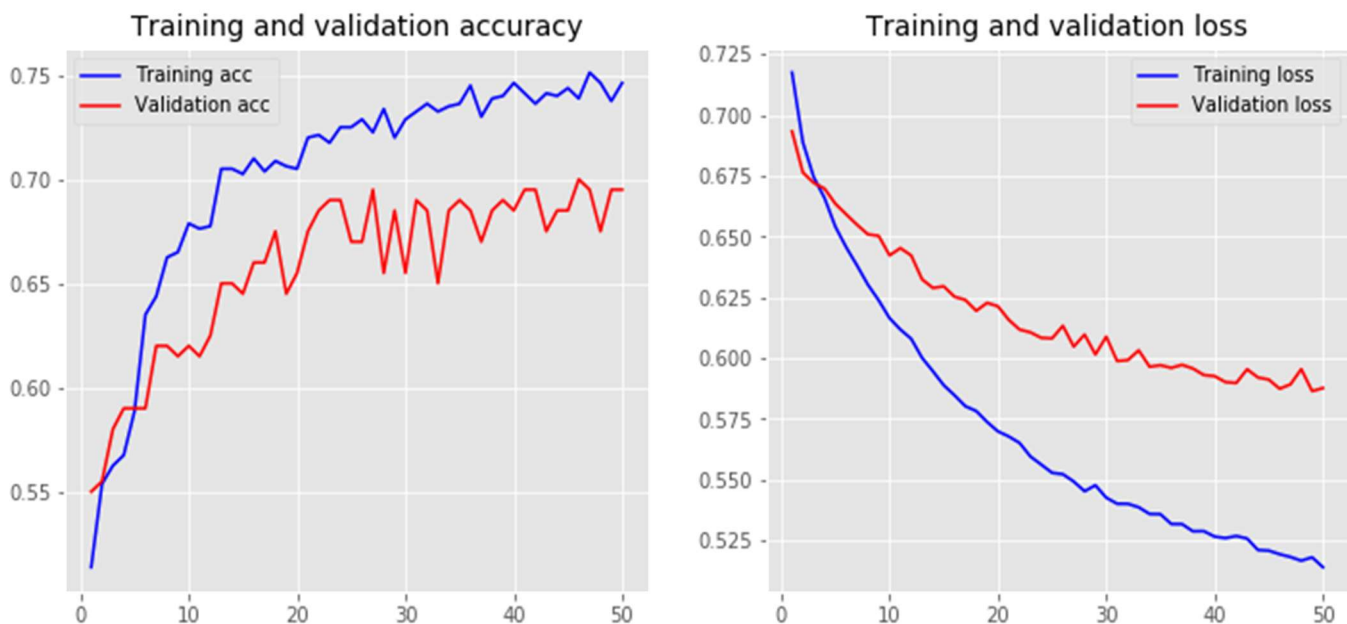
In the **Model Implementations and Training** phase of a Text Extraction project, the selected algorithms and techniques are put into practice to create functional models capable of extracting relevant information from text. This involves preparing the dataset by splitting it into training, validation, and test sets, allowing for effective learning and tuning of the model parameters. Training is conducted using various machine learning or deep learning frameworks, where the model learns to identify patterns and relationships within the data through iterative optimization processes. Hyperparameter tuning and regularization techniques are applied to enhance model performance and prevent overfitting, ensuring that the model generalizes well to unseen data.

4.4 MODEL EVALUATION METRICES

For **Model Evaluation Techniques**, several methods are employed to assess the model's performance and extraction accuracy. Common evaluation metrics include Precision, Recall, and F1-Score, which provide insights into the model's ability to correctly identify relevant entities while minimizing false positives and negatives. Additionally, confusion matrices are used to visualize performance across different classes or entities, facilitating a deeper understanding of strengths and weaknesses. Cross-validation techniques may also be applied to ensure that the model's performance is consistent across different subsets of the data. Finally, qualitative evaluations through manual review of extracted results help validate the model's effectiveness and guide any necessary adjustments before full deployment.

4.5 MODEL DEPLOYMENT: TESTING AND VALIDATION

ACCURACY AND LOSS GRAPHS:



4.6 WEB GUI'S DEVELOPMENT

```
from flask import Flask, request, render_template
import os

app = Flask(__name__)
UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Ensure the upload folder exists
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return "No file part"
    file = request.files['file']
    if file.filename == "":
        return "No selected file"
    file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
    file.save(file_path)

    # Placeholder for actual text extraction logic
    # extracted_text = extract_text(file_path)

    return f"File uploaded successfully! Processed text: [placeholder for extracted text]"
```

```

if __name__ == '__main__':
    app.run(debug=True)
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
    <title>Text Extraction Tool</title>
</head>
<body>
    <div class="container">
        <h1 class="mt-5">Text Extraction Tool</h1>
        <form action="/upload" method="post" enctype="multipart/form-data" class="mt-4">
            <div class="form-group">
                <label for="file">Upload Document</label>
                <input type="file" class="form-control-file" id="file" name="file" required>
            </div>
            <button type="submit" class="btn btn-primary">Extract Text</button>
        </form>
    </div>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>

```

/project-directory

```
├── app.py
└── templates
```

└─ index.html

- └─ uploads (create this folder manually)

Python app.py

4.7 RESULTS

OUTPUT 1

[illegible]

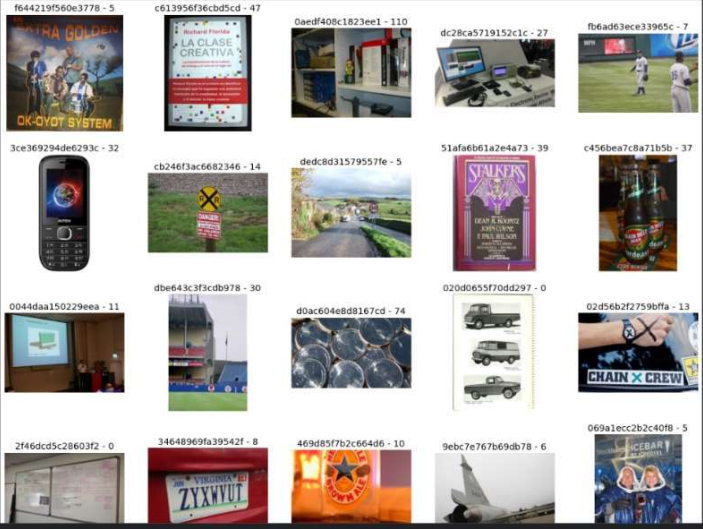
OUTPUT 2

Extracting Text from Images Draft saved

File Edit View Run Settings Add-ons Help

Display for first 25 images

```
[5]: fig, axs = plt.subplots(5, 5, figsize=(20, 20))
      axs = axs.flatten()
      for i in range(25):
          axs[i].imshow(plt.imread(img_fns[i]))
          axs[i].axis('off')
          image_id = img_fns[i].split('/')[-1].rstrip('.jpg')
          n_annot = len(annot.keys()[image_id == image_id])
          axs[i].set_title(f'{image_id} - {n_annot}')
      plt.show()
```



Right sidebar: Notebook

Input: + Add Input, Upload

DATASETS: textocr-text-extraction-from-images-ds

Output: /kaggle/working

Table of contents: Outline

Session options: ACCELERATOR: GPU P100, Quota: 06:00 / 30 hrs, LANGUAGE: Python, PERSISTENCE: No persistence, ENVIRONMENT: Pin to original environment (2022-06-21), INTERNET: Internet on, TAGS: Add Tags, Schedule a notebook to run, Code Help

OUTPUT 3

Extracting Text from Images Draft saved

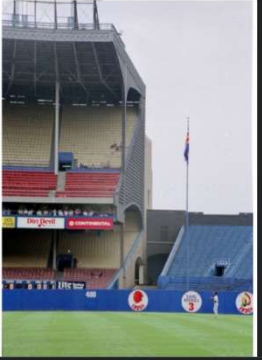
File Edit View Run Settings Add-ons Help

Method 1: pytesseract

```
[6]: import pytesseract
      # Example call
      print(pytesseract.image_to_string(img_fns[11], lang='eng'))
```

Shichi
ae : oe

```
[7]: fig, ax = plt.subplots(figsize=(10,10))
      ax.imshow(plt.imread(img_fns[11]))
      ax.axis('off')
      plt.show()
```



Right sidebar: Notebook

Input: + Add Input, Upload

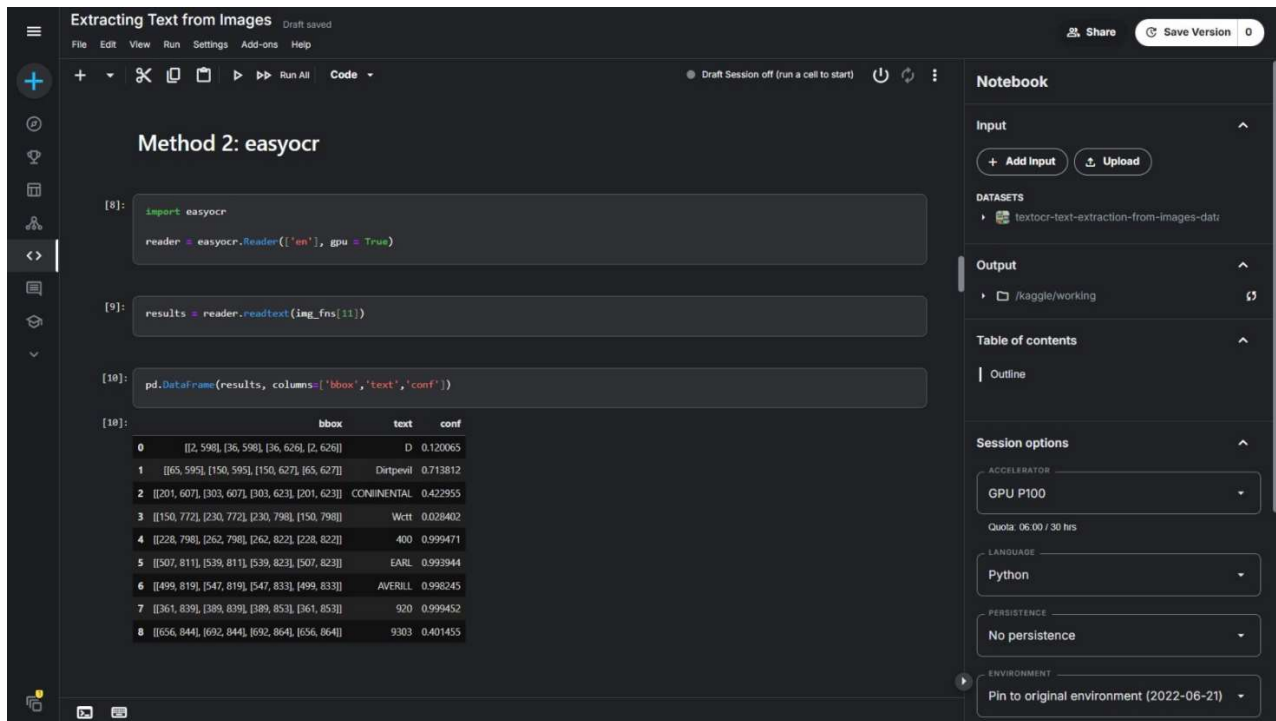
DATASETS: textocr-text-extraction-from-images-ds

Output: /kaggle/working

Table of contents: Outline

Session options: ACCELERATOR: GPU P100, Quota: 06:00 / 30 hrs, LANGUAGE: Python, PERSISTENCE: No persistence, ENVIRONMENT: Pin to original environment (2022-06-21), INTERNET: Internet on, TAGS: Add Tags

OUTPUT 4



Extracting Text from Images Draft saved

File Edit View Run Settings Add-ons Help

Method 2: easyocr

```
[8]: import easyocr
reader = easyocr.Reader(['en'], gpu = True)
```

```
[9]: results = reader.readtext(img_fns[11])
```

```
[10]: pd.DataFrame(results, columns=['bbox', 'text', 'conf'])
```

```
[10]:
```

	bbox	text	conf
0	[[2, 598], [36, 598], [36, 626], [2, 626]]	D	0.120065
1	[[65, 595], [150, 595], [150, 627], [65, 627]]	Dirtdevil	0.713812
2	[[201, 607], [303, 607], [303, 623], [201, 623]]	CONTINENTAL	0.422955
3	[[150, 772], [230, 772], [230, 798], [150, 798]]	Wett	0.028402
4	[[228, 798], [262, 798], [262, 822], [228, 822]]	400	0.999471
5	[[507, 811], [539, 811], [539, 823], [507, 823]]	EARL	0.993944
6	[[499, 819], [547, 819], [547, 833], [499, 833]]	AVERILL	0.998245
7	[[361, 839], [389, 839], [389, 853], [361, 853]]	920	0.999452
8	[[656, 844], [692, 844], [692, 864], [656, 864]]	9303	0.401455

Notebook

Input

+ Add Input Upload

DATASETS

textocr-text-extraction-from-images-data

Output

/kaggle/working

Table of contents

Outline

Session options

ACCELERATOR

GPU P100

Quota: 06:00 / 30 hrs

LANGUAGE

Python

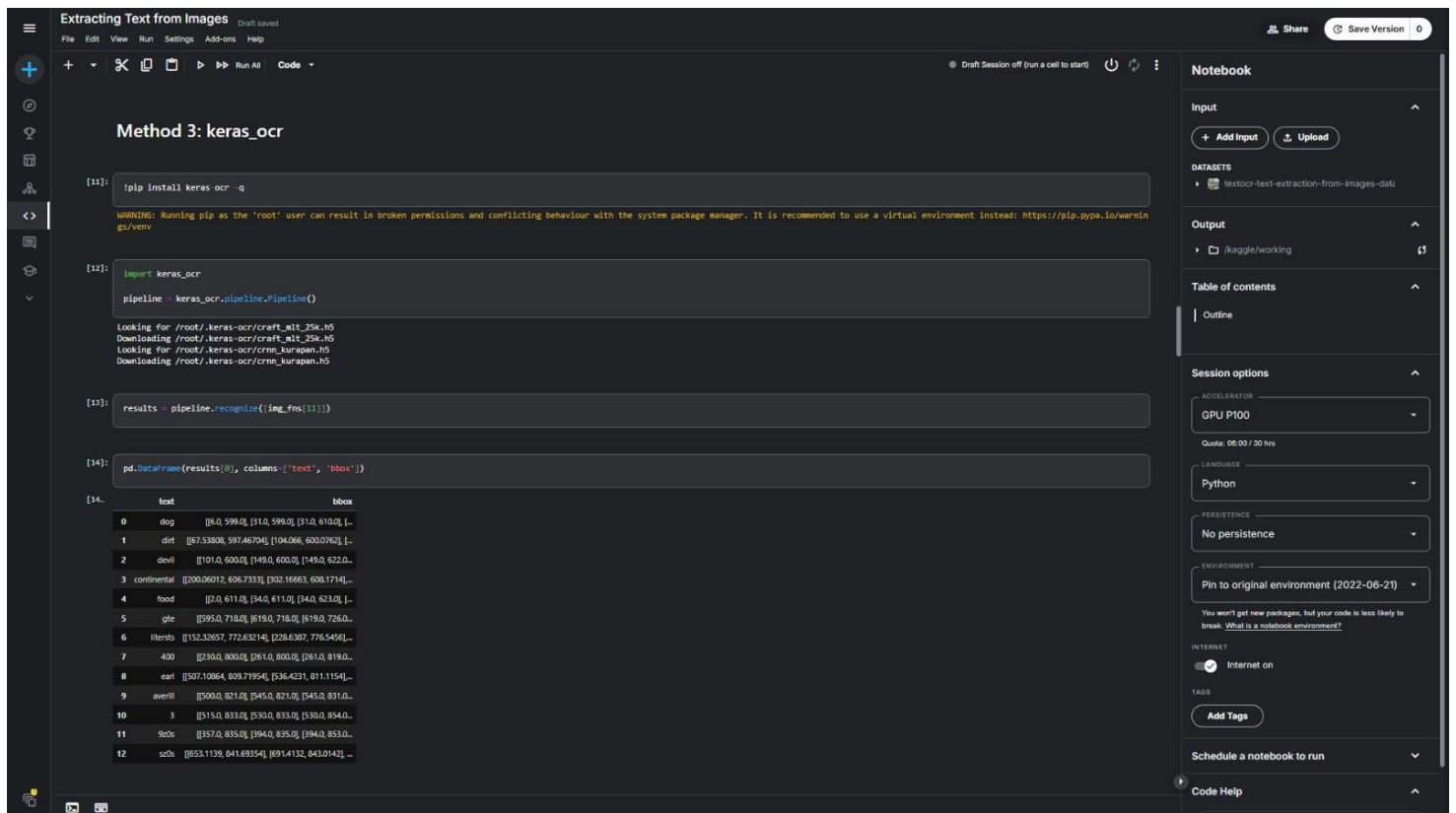
PERSISTENCE

No persistence

ENVIRONMENT

Pin to original environment (2022-06-21)

OUTPUT 5



Extracting Text from Images Draft saved

File Edit View Run Settings Add-ons Help

Method 3: keras_ocr

```
[11]: !pip install keras-ocr
```

WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: <https://pip.pypa.io/warnings/venv>

```
[12]: import keras_ocr
pipeline = keras_ocr.pipeline.Pipeline()
```

Looking for /root/.keras-ocr/craft_mit_25k.h5
Downloading /root/.keras-ocr/craft_mit_25k.h5
Looking for /root/.keras-ocr/crnn_kurapan.h5
Downloading /root/.keras-ocr/crnn_kurapan.h5

```
[13]: results = pipeline.recognize([img_fns[11]])
```

```
[14]: pd.DataFrame(results[0], columns=['text', 'bbox'])
```

```
[14]:
```

	text	bbox
0	dog	[[6, 599, 31, 599], [31, 610, 1, ...]]
1	dirt	[[67, 538, 597, 467], [104, 0, 600, 782], [...]]
2	devil	[[101, 600, 149, 600], [149, 622, 0, ...]]
3	continental	[[200, 601, 606, 733], [302, 1663, 606, 1714], [...]]
4	food	[[2, 611, 34, 611], [34, 623, 0, ...]]
5	gte	[[595, 718, 619, 718], [619, 718, 0, ...]]
6	illegals	[[152, 326, 772, 631], [228, 630, 776, 546], [...]]
7	400	[[230, 800, 261, 800], [261, 819, 0, ...]]
8	earl	[[507, 1084, 539, 795], [539, 421, 811, 1154], [...]]
9	averill	[[500, 821, 545, 821], [545, 831, 0, ...]]
10	3	[[515, 833, 530, 833], [530, 854, 0, ...]]
11	920c	[[37, 835, 394, 835], [394, 853, 0, ...]]
12	9303	[[653, 113, 691, 635], [691, 412, 843, 0142], [...]]

Notebook

Input

+ Add Input Upload

DATASETS

textocr-text-extraction-from-images-data

Output

/kaggle/working

Table of contents

Outline

Session options

ACCELERATOR

GPU P100

Quota: 06:00 / 30 hrs

LANGUAGE

Python

PERSISTENCE

No persistence

ENVIRONMENT

Pin to original environment (2022-06-21)

You won't get new packages, but your code is less likely to break. What is a notebook environment?

INTERNET

Internet on

TABS

Add Tags

Schedule a notebook to run

Code Help

OUTPUT 6

Extracting Text from Images

File Edit View Run Settings Add-ons Help

Run All Code

Draft Session off (run a cell to start)

6 itersts [[152.32657, 772.63214], [228.6367, 776.5456]...

7 400 [[250.0, 800.0], [261.0, 800.0], [261.0, 819.0]...

8 earl [[507.10664, 809.71954], [536.4231, 811.1154]...

9 averill [[500.0, 821.0], [545.0, 821.0], [545.0, 831.0]...

10 3 [[515.0, 833.0], [530.0, 833.0], [530.0, 854.0]...

11 9z0s [[357.0, 835.0], [394.0, 835.0], [394.0, 853.0]...

12 sz0s [[853.1139, 841.69354], [691.4132, 843.0142]...

[15]:

fig, ax = plt.subplots(figsize (16, 16))
keras_ocr.tools.drawAnnotations(plt.imread(img_fns[11]), results[0], ax=ax)
ax.set_title('Keras OCR Result Example')
plt.show()

Keras OCR Result Example

Notebook

Input

+ Add Input Upload

DATASETS

textocr-text-extraction-from-images-data

Output

/kaggle/working

Table of contents

Outline

Session options

ACCELERATOR

GPU P100

Quota: 06:00 / 30 hrs

LANGUAGE

Python

PERSISTENCE

No persistence

ENVIRONMENT

Pin to original environment (2022-06-21)

You won't get new packages, but your code is less likely to break. What is a notebook environment?

INTERNET

Internet on

TABS

Add Tags

5.CONCLUSION

5.1 PROJECT CONCLUSION

In conclusion, the Text Extraction project successfully developed a robust system that automates the identification and extraction of relevant information from various document types, enhancing data accessibility and usability. By leveraging advanced techniques in Natural Language Processing (NLP) and Optical Character Recognition (OCR), the system can accurately process unstructured data from formats such as PDFs, images, and web content. The user-friendly web interface facilitates seamless interaction, allowing users to upload documents and retrieve extracted information efficiently. Throughout the project, extensive testing and evaluation ensured that the system meets accuracy and performance standards, demonstrating its potential to significantly reduce manual data entry efforts and streamline workflows across various applications.

5.2 FUTURE ENHANCEMENT

Looking ahead, several enhancements can be made to further improve the functionality and user experience of the Text Extraction system. First, expanding the range of document types and formats supported by the system will enhance its versatility, accommodating more user needs. Implementing machine learning models that continuously learn from user interactions and feedback can lead to improved extraction accuracy over time. Additionally, integrating more advanced features, such as multi-language support and sentiment analysis, would broaden the applicability of the system to international markets and diverse datasets.

Another potential enhancement is the incorporation of real-time data processing capabilities, enabling users to receive immediate feedback on their uploaded documents. Furthermore, improving the system's scalability by deploying it on cloud platforms will allow it to handle larger volumes of data without compromising performance. Lastly, developing mobile applications for on-the-go access to the text extraction services would cater to a broader audience and align with the growing demand for mobile solutions. By pursuing these enhancements, the Text Extraction project can evolve into a more powerful tool that meets the increasing demands of users in an ever-changing data landscape.