

Optical Character Recognition

Introduction:

The objective of an Optical Character Recognition (OCR) project is to develop a software or system that can accurately recognize and interpret text characters from images, scanned documents, or printed materials, and convert them into editable and searchable digital formats. The main goal of OCR is to automate the process of data entry and eliminate the need for manual transcription, thus improving efficiency and reducing errors. The project may also aim to enhance OCR's ability to recognize handwriting, fonts, languages, and complex layouts, and integrate it with other applications such as document management, translation, and text-to-speech technologies.

Abstract:

Optical Character Recognition (OCR) is a technology that enables the conversion of printed or handwritten text into machine-encoded characters that can be easily processed and edited by computers. The objective of this project is to develop an OCR system that can accurately recognize and interpret text characters from various sources, including images, scanned documents, and printed materials. The project involves several stages, including image preprocessing, feature extraction, and classification, where machine learning algorithms and techniques are utilized to identify and extract meaningful patterns and features from the input data.

The proposed OCR system is designed to be highly efficient and accurate, with the capability to handle a wide range of text styles, fonts, and languages. The system's performance will be evaluated using a standard dataset and compared to existing OCR solutions to validate its effectiveness and reliability. Additionally, the project aims to integrate the OCR system with other applications, such as document management, translation, and speech-to-text technologies, to enhance its functionality and usability.

The results of this project are expected to provide a useful tool for automating the data entry process, improving the accuracy and efficiency of document processing, and reducing manual transcription errors. This technology has the potential to benefit a wide range of industries, including healthcare, finance, and legal services, where large volumes of documents need to be processed and managed accurately and quickly.

Software Requirements:

- Python Version 3.11.3
- IDE – Recommended [PyCharm / Python-IDLE]
- Tesseract – Executable File Application
- Tesseract Library Files – pytesseract

Algorithm / Work Flow of Tesseract:

Pytesseract is a Python wrapper for the Tesseract OCR engine, which provides an easy-to-use interface for performing OCR tasks on images and text data. Here is a high-level overview of the algorithm design of Pytesseract:

1. Input data preprocessing: Pytesseract accepts input data in various formats, including images, PDF files, and raw text. The input data is first preprocessed to ensure that it is in a suitable format for OCR, such as converting images to grayscale or PDF files to images.

2. Text extraction: The preprocessed input data is then passed to the Tesseract OCR engine through the pytesseract API. The engine applies a series of image processing and feature extraction techniques, such as binarization, segmentation, and character recognition, to extract the text from the input data.

3. Post-processing: The extracted text is further processed to remove any noise, punctuation, or unwanted characters that may have been recognized incorrectly. This may involve using regular expressions, spell-checking algorithms, or language models to improve the accuracy and quality of the output text.

4. Output formatting: The final output of Pytesseract is the recognized text in a standardized format, such as plain text or HTML. The output can be further customized based on the user's preferences, such as specifying the language, font, or OCR mode to use.

5. Error handling and logging: Pytesseract includes robust error handling and logging mechanisms to ensure that any errors or exceptions that occur during the OCR process are handled gracefully and logged appropriately. This helps to diagnose and resolve any issues that may arise during OCR and improve the overall reliability and performance of the algorithm.

Overall, the algorithm design of Pytesseract is a combination of image processing, feature extraction, and post-processing techniques that work together to accurately recognize text from input data. The algorithm is highly configurable and customizable, allowing users to tailor it to their specific OCR needs and requirements.

Pre – Required Installations:

1.Install Tesseract - OCR Executable Link for Downloading Tesseract - <https://digi.bib.uni-mannheim.de/tesseract/tesseract-ocr-w64-setup-5.3.1.20230401.exe>

2.Install pytesseract Libraries through command prompt - pip install pytesseract

Steps to deploy the project :

Step 1 : Install Tesseract - OCR Executable file using the above link.

Step 2 : Import pytesseract Libraries for using in the program.

Step 3 : Save the images (which needs to be recognized) in the same folder where the program file is placed

Step 4 : Enter the image file name which needs to be recognized in line 17

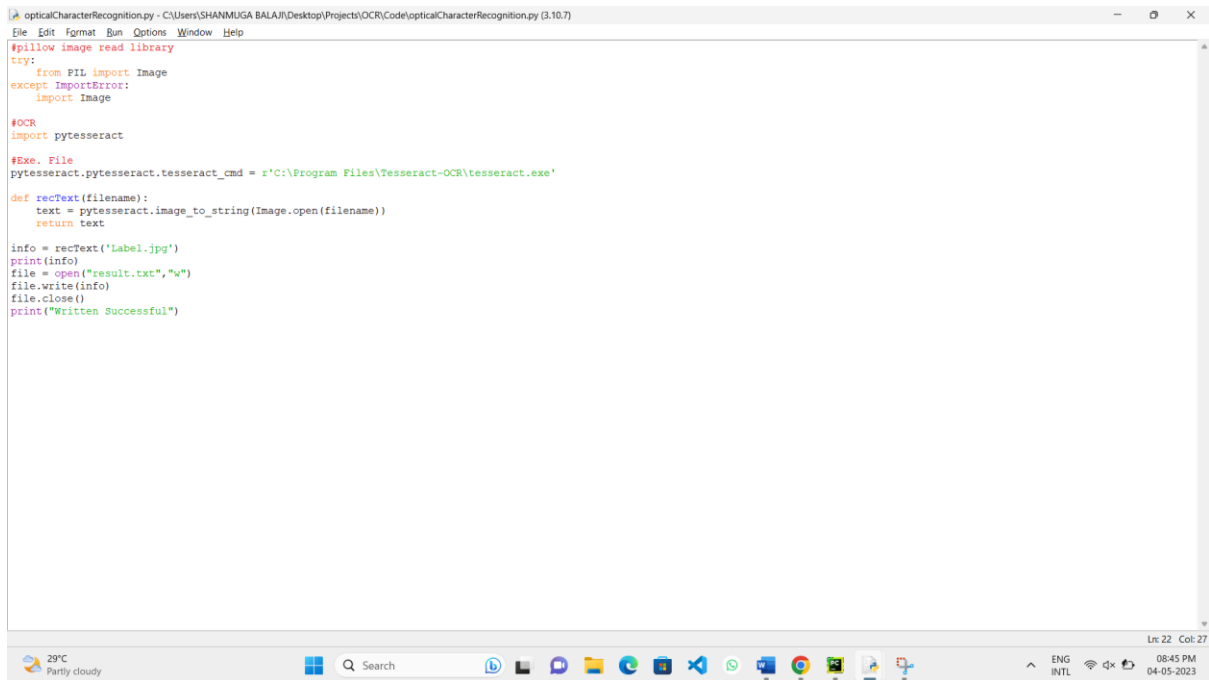
Step 5 : Now, Run the program

The output/predicted values will be stored in the text document which is named as result

Source code:

```
try:
    from PIL import Image
except ImportError:
    import Image
#OCR
import pytesseract
#Exe. File
pytesseract.pytesseract.tesseract_cmd =
r'C:\Program Files\Tesseract-OCR\tesseract.exe'
def recText(filename):
    text =
pytesseract.image_to_string(Image.open(filename))
    return text
info = recText('Label.jpg')
print(info)
file = open("result.txt","w")
file.write(info)
file.close()
print("Written Successful")
```

Source Code Images:



```
opticalCharacterRecognition.py - C:\Users\SHANMUGA BALAJI\Desktop\Projects\OCR\Code\opticalCharacterRecognition.py (3.10.7)
File Edit Format Run Options Window Help
#pillow image read library
try:
    from PIL import Image
except ImportError:
    import Image

#OCR
import pytesseract

#Exe. File
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

def recText(filename):
    text = pytesseract.image_to_string(Image.open(filename))
    return text

info = recText('Label.jpg')
print(info)
file = open("result.txt", "w")
file.write(info)
file.close()
print("Written Successful")
```

```
#pillow image read library
try:
    from PIL import Image
except ImportError:
    import Image

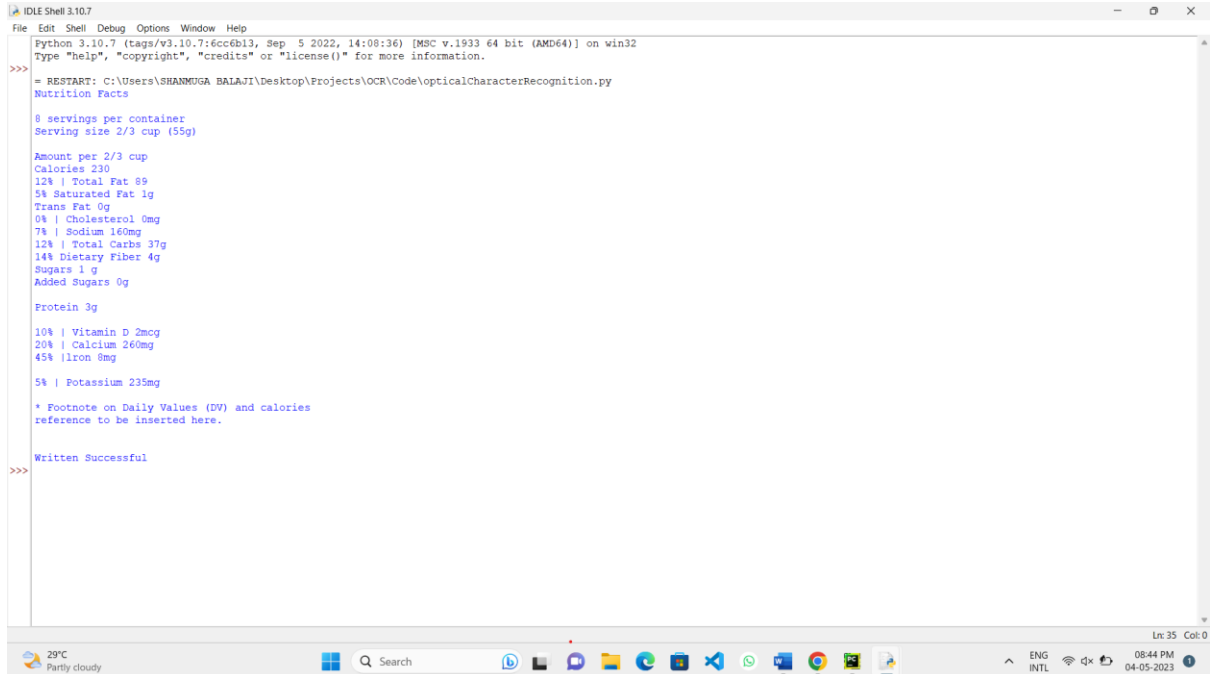
#OCR
import pytesseract

#Exe. File
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

def recText(filename):
    text = pytesseract.image_to_string(Image.open(filename))
    return text

info = recText('Label.jpg')
print(info)
file = open("result.txt", "w")
file.write(info)
file.close()
print("Written Successful")
```

Outputs:



```
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\SHANMUGA BALAJI\Desktop\Projects\OCR\Code\opticalCharacterRecognition.py
Nutrition Facts

8 servings per container
Serving size 2/3 cup (55g)

Amount per 2/3 cup
Calories 230
12% | Total Fat 8g
5% Saturated Fat 1g
Trans Fat 0g
0% | Cholesterol 0mg
7% | Sodium 160mg
12% | Total Carbs 37g
14% Dietary Fiber 4g
Sugars 1 g
Added Sugars 0g

Protein 3g

10% | Vitamin D 2mcg
20% | Calcium 260mg
45% |Iron 8mg

5% | Potassium 235mg

* Footnote on Daily Values (DV) and calories
reference to be inserted here.

Written Successful
>>>
```

```
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: C:\Users\SHANMUGA BALAJI\Desktop\Projects\OCR\Code\opticalCharacterRecognition.py
Nutrition Facts

8 servings per container
Serving size 2/3 cup (55g)

Amount per 2/3 cup
Calories 230
12% | Total Fat 8g
5% Saturated Fat 1g
Trans Fat 0g
0% | Cholesterol 0mg
7% | Sodium 160mg
12% | Total Carbs 37g
14% Dietary Fiber 4g
Sugars 1 g
Added Sugars 0g

Protein 3g

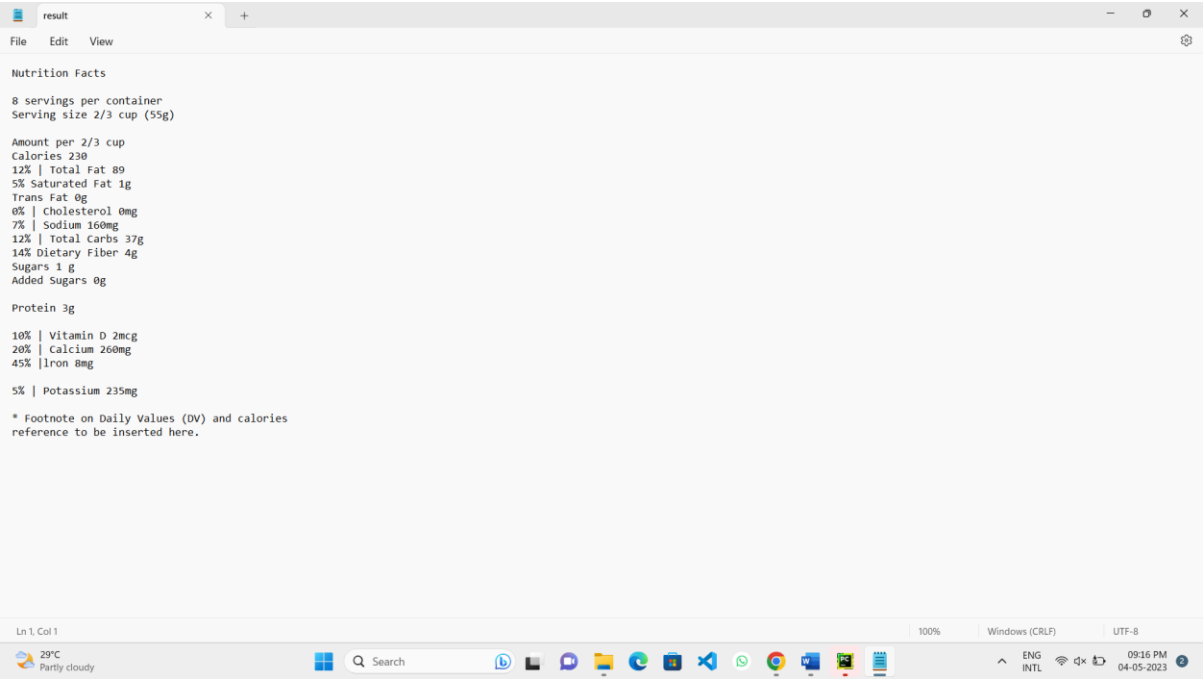
10% | Vitamin D 2mcg
20% | Calcium 260mg
45% |Iron 8mg

5% | Potassium 235mg

* Footnote on Daily Values (DV) and calories
reference to be inserted here.

Written Successful
```

Conversion of Image to Text:



Input

Nutrition Facts	
8 servings per container	
Serving size 2/3 cup (55g)	
Amount per 2/3 cup	
Calories 230	
% DV*	
12%	Total Fat 8g
5%	Saturated Fat 1g
	Trans Fat 0g
0%	Cholesterol 0mg
7%	Sodium 160mg
12%	Total Carbs 37g
14%	Dietary Fiber 4g
	Sugars 1g
	Added Sugars 0g
	Protein 3g
10%	Vitamin D 2mcg
20%	Calcium 260mg
45%	Iron 8mg
5%	Potassium 235mg
* Footnote on Daily Values (DV) and calories reference to be inserted here.	

Output

Nutrition Facts

8 servings per container
Serving size 2/3 cup (55g)

Amount per 2/3 cup
Calories 230
12% | Total Fat 8g
5% Saturated Fat 1g
Trans Fat 0g
0% | Cholesterol 0mg
7% | Sodium 160mg
12% | Total Carbs 37g
14% Dietary Fiber 4g
Sugars 1 g
Added Sugars 0g

Protein 3g

10% | Vitamin D 2mcg
20% | Calcium 260mg
45% | Iron 8mg

5% | Potassium 235mg

* Footnote on Daily Values (DV) and calories reference to be inserted here.

Limitations / Constraints:

Optical character recognition (OCR) technology has made great strides in recent years, but there are still several constraints and limitations to be aware of. Here are some of the most significant ones:

1. **Quality of the source document:** The quality of the source document can greatly affect OCR accuracy. If the document is blurry, has low contrast, or contains noise, it can make it difficult for the OCR system to accurately recognize characters.
2. **Font and formatting:** Different fonts and formatting styles can pose a challenge for OCR systems, especially if they are highly stylized or have unique characters.
3. **Language:** OCR systems are typically designed to recognize characters from specific languages. If the language in the source document is different from what the OCR system was trained on, the accuracy of the system can be significantly impacted.
4. **Handwriting:** OCR systems are generally not very accurate at recognizing handwriting. This is because handwriting can vary greatly between individuals, and the system may not be able to correctly interpret the individual style of each writer.
5. **Layout and structure of the document:** The layout and structure of the document can also impact OCR accuracy. For example, if the text is in multiple columns or if there are graphics or images on the page, it can make it more difficult for the OCR system to accurately recognize characters.

Results /Conclusions:

In conclusion, optical character recognition (OCR) technology has come a long way in recent years and has proved to be a valuable tool in digitizing text and making it searchable. However, despite significant advancements, OCR systems still face constraints and limitations, such as the quality of the source document, font and formatting, language, handwriting, layout, and processing time. It is important to keep these limitations in mind when designing and implementing OCR projects, and to carefully assess the accuracy and feasibility of using OCR for specific applications.

Overall, OCR technology offers great potential for improving efficiency and accessibility in various industries and applications, but its limitations should be carefully considered to ensure optimal results. A reliable project of OCR system that can enhance efficiency and accessibility in various applications is developed successfully.

References:

Here are some references that you might find useful for an OCR project:

1. Tesseract OCR Engine: <https://github.com/tesseract-ocr/tesseract>
2. Pytesseract: <https://pypi.org/project/pytesseract/>