Auto-Entry

Prototype development and financial modelling

Submitted by – Utkarsh Balooni

Step – 1: Prototype Selection

Abstract:

Data entry is vital for facilitating smooth operational management in any business. However, the manual process of data entry is time-consuming, resource intensive and prone to error. This leads to reduced efficiency and lower profitability.

To tackle this problem, the data entry process can be automated using the technique of Optical Character Recognition. This project aims to develop a solution that can automatically extract relevant data from documents (invoices, forms, IDs) and insert it into an appropriate database. The model developed would be robust and capable of effectively processing diverse document formats, layouts, and languages.

By streamlining the data entry process and minimizing human effort, the objective is to boost productivity and eliminate errors in data management. Eventually, this will improve overall business operations and will lead to higher profits.

Feasibility:

There are many state-of-the-art technologies available for Optical Character Recognition. Google cloud vision, Amazon Textract, ABBY FineReader and many more. The accuracy and reliability of the service will depend on the underlying model used.

The feasibility also relies on the quality of the source documents. OCR works best with clear, well-scanned or photographed documents. If the source documents are of low quality, the accuracy of the extracted data may suffer. Also, there are several OCR APIs and SDKs available that can integrate into the service for effective data entry automation

Viability:

The viability of developing an automated data entry service is promising. As businesses continue to face challenges in handling large volumes of physical and digital documents, the demand for efficient data entry solutions is on the rise. By

leveraging OCR technology, the service can accurately extract text from various sources, including printed documents, images, and scanned files. This capability can significantly reduce manual data entry efforts and potential errors, leading to increased productivity and improved data accuracy.

The viability of the service further depends on the availability of reliable OCR technologies that can handle different languages, document formats, and varying image qualities.

Monetization:

There are several monetization strategies that could be considered:

- **Subscription Model:** Offer different subscription tiers based on usage levels or features. For example, charge more for higher volume processing or priority support.
- **Pay-per-use:** Implement a pay-per-use model where customers pay for each OCR transaction or data extraction.

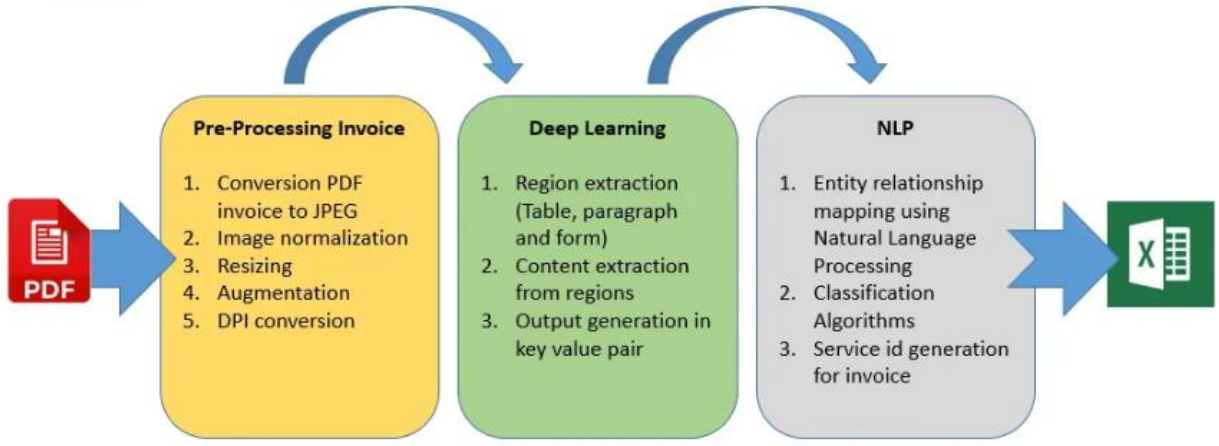
- **Integration Fees:** Charge additional fees for custom integration with clients' existing systems or third-party applications.
- **API Usage:** Provide an API for developers to integrate OCR capabilities into their own applications, consider charging based on API usage.
- **Value-added Services:** Offer additional services like data validation, data cleaning, or data analytics, which customers may be willing to pay extra for.

Step – 2: Prototype Development

Major use cases of data entry are:

- Number plates
- Legal Documents
- Text Extraction
- Banking
- Invoices
- Healthcare

We will develop a small-scale prototype for the invoice service. The code can be tailored to meet the requirements for all the use cases.



Pipeline Architecture

Pre-Processing invoice:

The first step is preparing the dataset for training and testing. We collect images of invoices and convert them to a suitable format.

Firstly, we import the necessary libraries.

```
from PIL import Image
import numpy as np
```

- **Image Normalization:** Image normalization technique is used to change the range of pixel intensity values in order to improve contrast of the image. Histogram Equalized or Contrast

Stretched mechanisms are widely used for image normalization. In this use-case we have employed Contrast Stretched technique.

```
def image_normalization(image_path):
    image = Image.open(image_path)
    normalized_image = np.array(image) / 255.0  # Normalize to [0, 1]
    return normalized_image
```

- **Image Resize:** We are training the deep neural network with size 600x600 pixel of images. Resize function of PIL python module have been used for resizing the images.

```
def resize_image(image_path, new_width, new_height):
    image = Image.open(image_path)
    resized_image = image.resize((new_width, new_height), Image.LANCZOS)
    return resized_image
```

- **Image Conversion:** All the images converted to RGB (channel 3) format and encoded to JEPG.

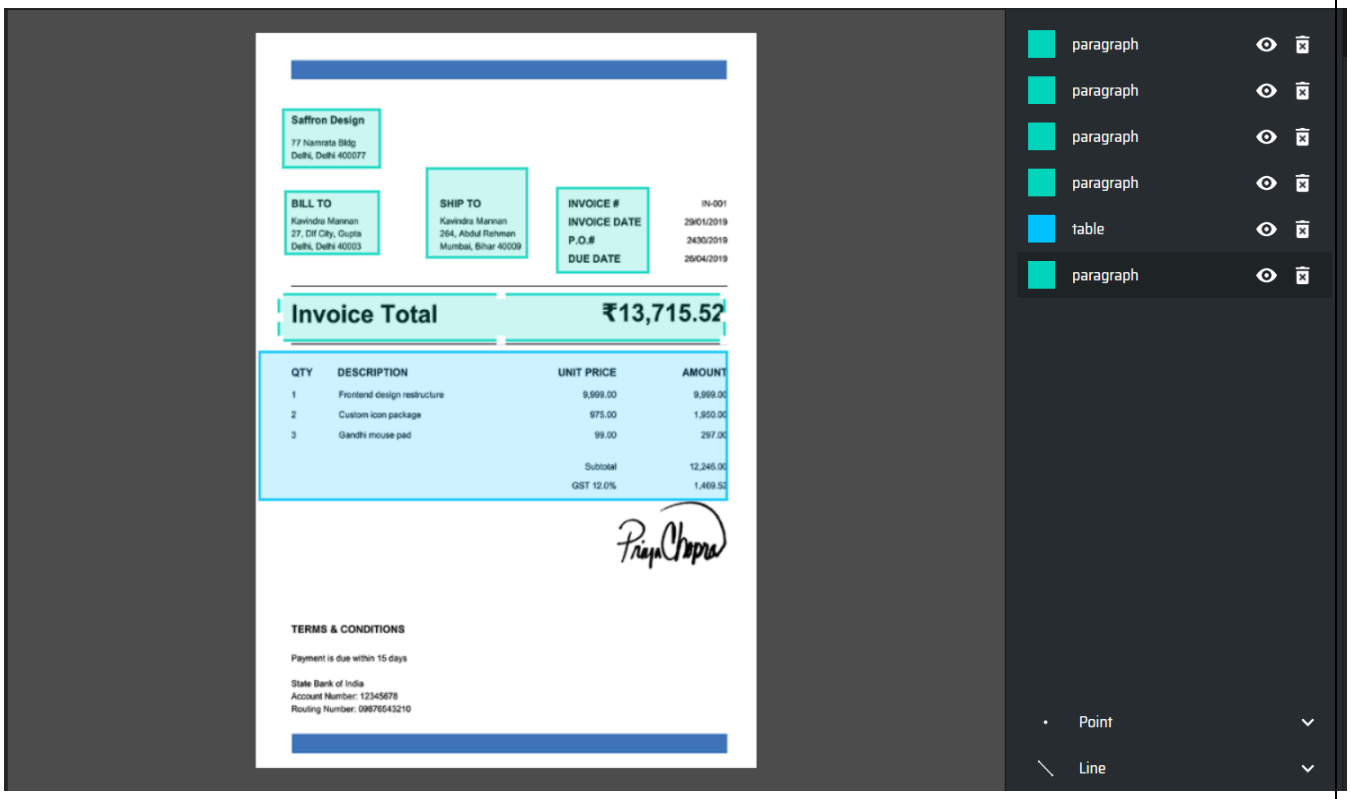
```
def convert_image(image_path, output_format='JPEG'):
    image = Image.open(image_path)
    converted_image = image.convert(output_format)
    return converted_image
```

- **Dots per Inch (DPI) Conversion:** We will work on images with high intensity, to accrue the same all the images converted to 300 DPI.

```
def convert_dpi(image_path, new_dpi):
    image = Image.open(image_path)
    dpi_width, dpi_height = image.info['dpi']
    aspect_ratio = dpi_height / dpi_width

    new_width = int(image.width * new_dpi / dpi_width)
    new_height = int(new_width * aspect_ratio)

    resized_image = image.resize((new_width, new_height), Image.LANCZOS)
    resized_image.info['dpi'] = (new_dpi, new_dpi)

    return resized_image
```

Image Labelling:

Images are manually annotated into 3 categories: paragraphs, tables and forms using make sense ai. (an online image annotation tool)



The annotation data (image name, bounding box co-ordinates, height and width) are stored into a CSV file and used for tensor flow record creation later.

label_name	bbox_x	bbox_y	bbox_width	bbox_height	image_name	image_width	image_height
paragraph	28	76	99	57	Screenshot 2023-07-17 231548.png	512	730
paragraph	30	157	95	62	Screenshot 2023-07-17 231548.png	512	730
paragraph	175	134	102	88	Screenshot 2023-07-17 231548.png	512	730
paragraph	308	154	93	84	Screenshot 2023-07-17 231548.png	512	730
table	4	316	477	147	Screenshot 2023-07-17 231548.png	512	730
paragraph	24	259	455	46	Screenshot 2023-07-17 231548.png	512	730

Image Augmentation:

In order to achieve a good performance in a deep network, we need large amounts of data for training. The python library imgaug is used for this

purpose. The augmenters used are affine translation, brightness, gaussian blur, horizontal flip.

```
import imgaug.augmenters as iaa
import imageio

def augment_single_image(input_image_path, output_image_path, augment_factor=5):
    # Read the input image
    image = imageio.imread(input_image_path)

    # Create an augmentation sequence
    seq = iaa.Sequential([
        iaa.Affine(translate_px={"x": (-20, 20), "y": (-20, 20)}),  # Affine
            translation
        iaa.MultiplyBrightness((0.8, 1.2)),  # Brightness adjustment
        iaa.GaussianBlur(sigma=(0.0, 3.0)),  # Gaussian blur
        iaa.Fliplr(0.5)  # Horizontal flip
    ])

    # Augment the image and save the augmented images
    for i in range(augment_factor):
        augmented_image = seq(image=image)
        output_path = f"{output_image_path}_augmented_{i}.png"
        imageio.imwrite(output_path, augmented_image)
```

Model Training:

We will use a convolutional neural network to train the model.

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.utils import to_categorical

IMG_WIDTH, IMG_HEIGHT = 150,150  # Target size for images after resizing
BATCH_SIZE = 32  # Number of images processed together in each training batch
EPOCHS = 10  # Number of training epochs
TRAIN_DIR = "D:\Feynn labs"
LABELS_FILE = "D:\Feynn labs"
```

```
def load_dataset(data_dir, labels, target_size):
    images = []
    label_values = []
    for image_file, label in labels.items():
        image_path = os.path.join(data_dir, image_file)
        image = np.load(image_path)
        images.append(image)
        label_values.append(label)
    return np.array(images), np.array(label_values)

def create_model(input_shape, num_classes):
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(128, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics
        =['accuracy'])
    return model
```

```
def main():
    labels = load_labels(LABELS_FILE)

    # Create a mapping of label classes to integer labels
    class_to_int = {label: idx for idx, label in enumerate(set(labels.values
        ()))}

    # Convert label strings to integers
    int_labels = [class_to_int[label] for label in labels.values()]

    # One-hot encode the integer labels
    num_classes = len(set(int_labels))
    one_hot_labels = to_categorical(int_labels, num_classes=num_classes)


    images, _ = load_dataset(TRAIN_DIR, labels, (IMG_WIDTH, IMG_HEIGHT, 1))


    num_samples = len(images)
    split_idx = int(0.8 * num_samples)  # 80% for training, 20% for validation
    x_train, y_train = images[:split_idx], one_hot_labels[:split_idx]
    x_val, y_val = images[split_idx:], one_hot_labels[split_idx:]

    # Create the CNN model
    input_shape = (IMG_WIDTH, IMG_HEIGHT, 1)
    model = create_model(input_shape, num_classes)
```

```
    # Train the model
    model.fit(x_train, y_train, batch_size=BATCH_SIZE, epochs=EPOCHS,
        validation_data=(x_val, y_val))

    # Save the trained model
    model.save("trained_model.h5")

if __name__ == "__main__":
    main()
```

The model is optimized by tuning the hyper-parameters.

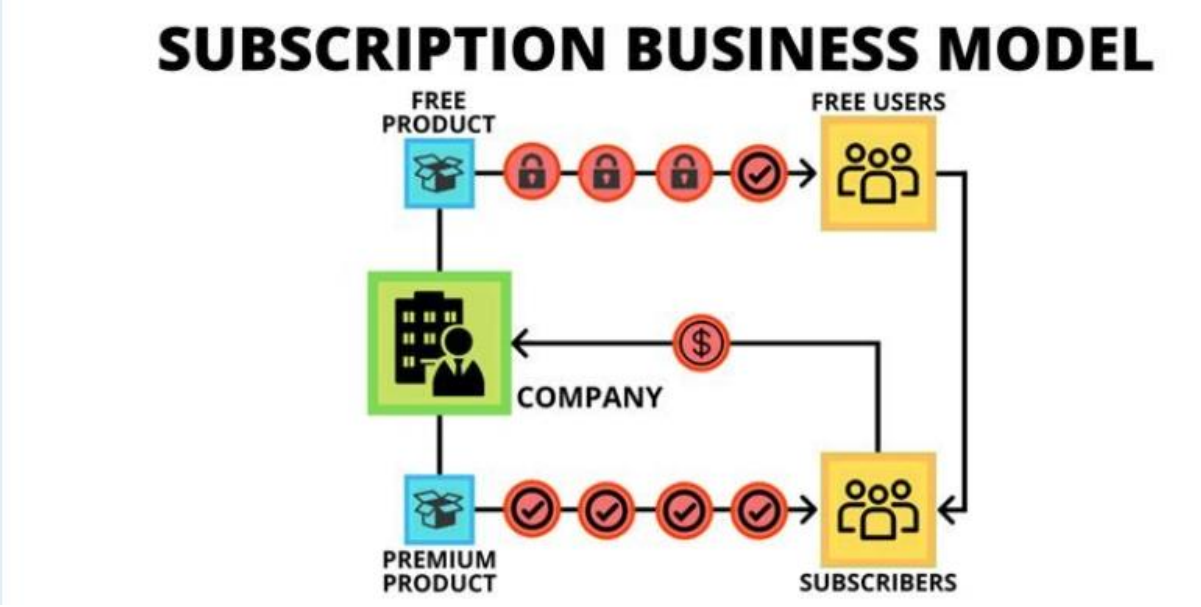
Text Extraction:

Once we have the segmented images, the next task is to extract text from them. This is done using tesseract library of python.

```
import cv2
import numpy as np
import pytesseract

invoice_image_path = "D:\Feynn labs"

# Preprocess the image using OpenCV
def preprocess_image(image_path):
    image = cv2.imread(image_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    return gray
preprocessed_image = preprocess_image(invoice_image_path)

# Perform text extraction using pytesseract
extracted_text = pytesseract.image_to_string(preprocessed_image)
```

Step – 3: Business Modelling

The most profitable strategy would be to use a subscription model and the ideal target segment would be small/large businesses.



- **Service offering:** Various subscription plans can be offered for the OCR service. Depending on the target market and use cases, different tiers of plans can be set up.

  Basic Plan: Suitable for individual users or small businesses with limited OCR needs.

  Pro Plan: Catered to medium-sized businesses with higher usage and additional features.

  Enterprise Plan: Designed for large organizations with extensive OCR requirements and premium support.

- **Pricing Strategy:** Appropriate pricing needs to be set by analysing pricing data for other competing services. Flexible billing options and occasional discounts should be accounted for. A free trial of the service could be offered for increasing the customer base.

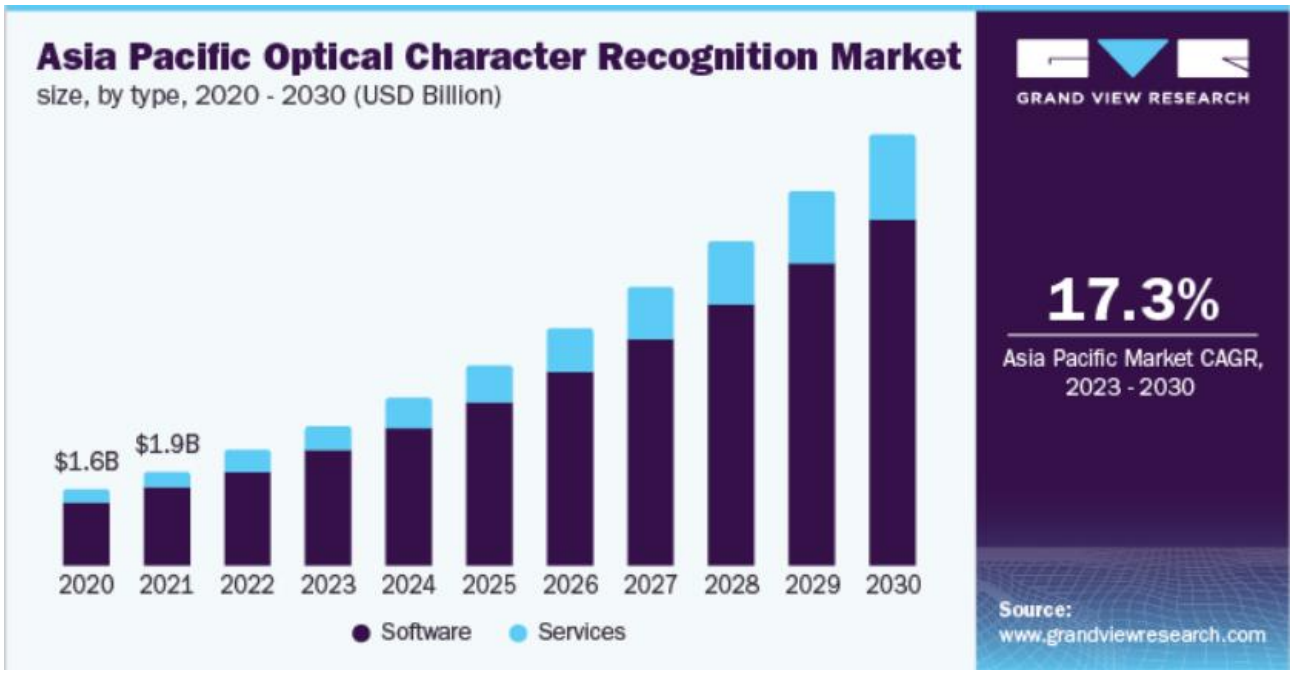
- **Service Development:** A dedicated team of ML engineers and web developers is needed to build, implement and maintain the service at a large scale. This requirement can be mitigated for small scale implementation of the service.

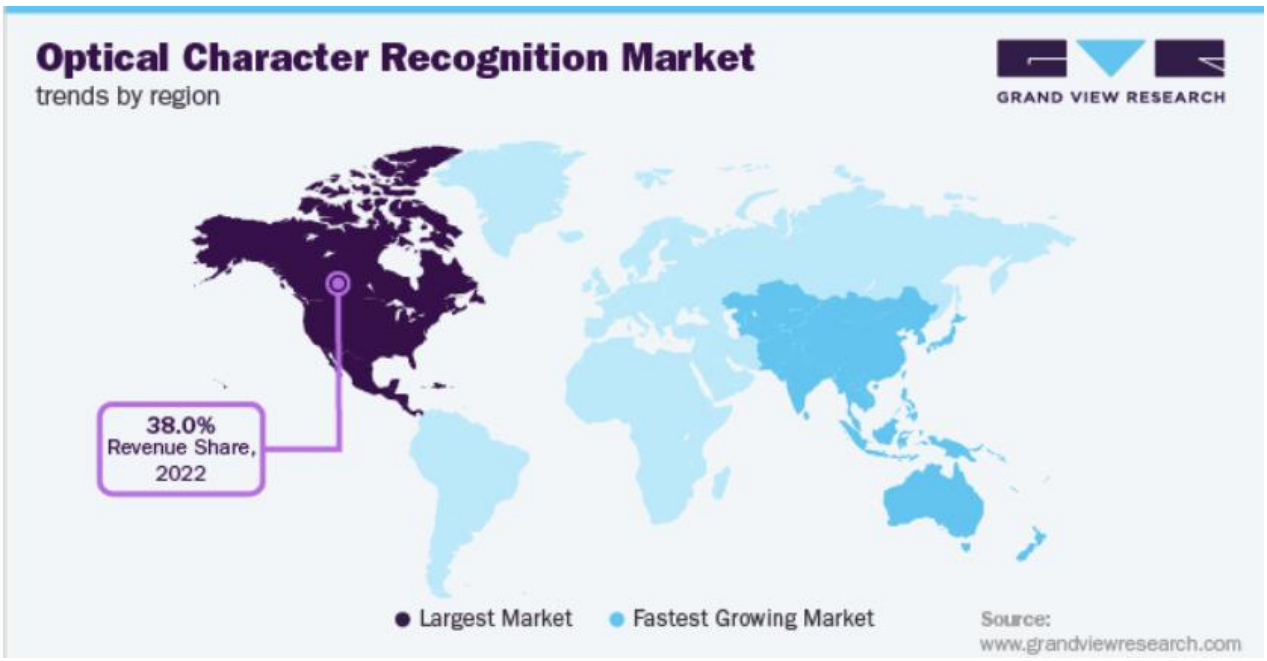
- **Customer Support:** Dedicated customer support should be offered to subscribers 24/7. Responsive assistance via email, whatsapp, phone can be provided.

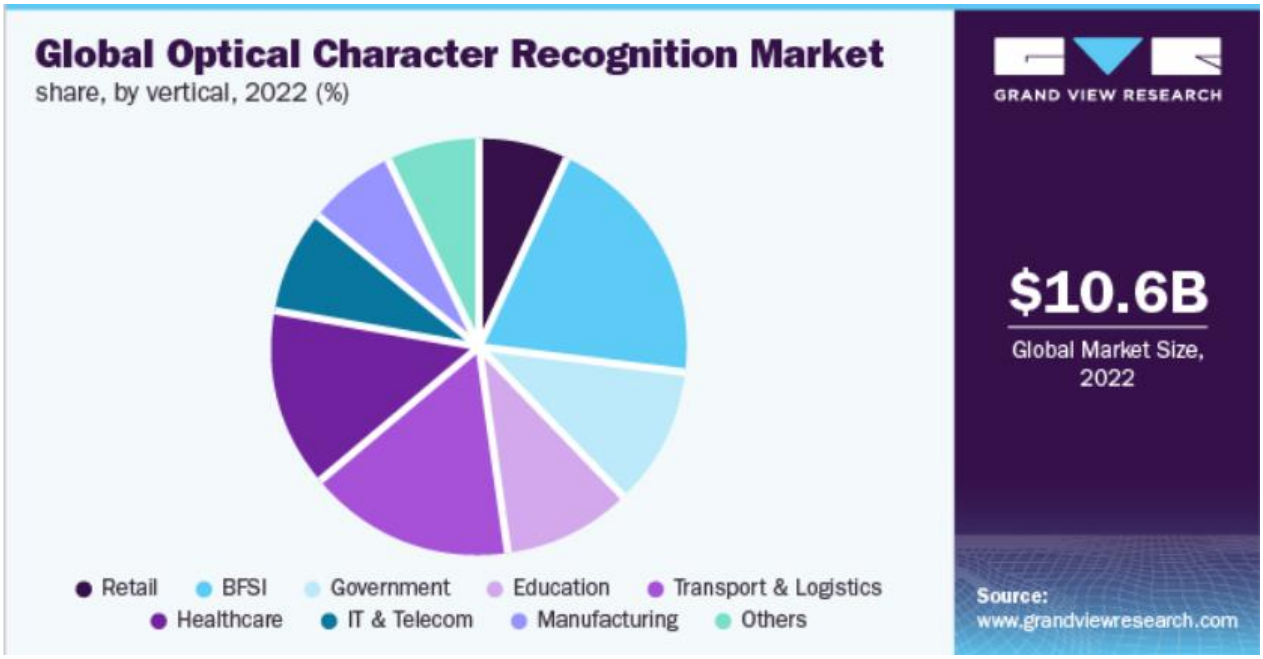
- **Marketing and Sales:** Continuous efforts should be put in to increase the reach of the product. Advertisements, referral programs, special promotions can be utilized for this task.

Step – 4: Financial modelling (equation)

The global optical character recognition market size was valued at USD 10.62 billion in 2022 and is expected to expand at a compound annual growth rate (CAGR) of 14.8% from 2023 to 2030.







Some of the prominent players in the global optical character recognition market include:

- ABBYY.
- Adobe.

- Anyline GmbH
- ATAPY Software
- Captricity Inc.
- Creaceed S.P.R.L.
- CVISION Technologies, Inc.
- Exper-OCR, Inc.
- Google LLC
- International Business Machines Corporation
- IntSig Information Co., Ltd. Corporation
- IRIS S.A.
- LEAD Technologies, Inc.
- Microsoft
- NAVER Corp.
- Nuance Communications, Inc.
- Open Text Corporation

Financial equation:

Suppose

y = Total yearly profit

m = Pricing of the product (yearly)

x(t) = Total yearly sales of the product (as a function of time)

c = yearly maintenance cost of the service

The financial equation is then given by:

$y = m*x(t) – c$

Calculating total cost:

Taking some valid assumptions, the yearly cost can be estimated as follows.

- Development cost – Hiring an ML engineer for development would cost around ₹8,00,000 per year.
- Infrastructure cost – All cloud storage and ocr service subscription would cost around ₹2,20,000 yearly.
- Marketing cost – Ad campaigns, partnerships would cost around ₹1,20,000 per year.
- Customer support – Hiring and training would cost around ₹5,00,000 per year.
- Miscellaneous cost – Buffer cost (operations, CAC etc.) around ₹50,000 per year.

Total yearly cost for the product – ₹16,90,000

Calculating x(t):

Since the market trend is linear, we will use 2-point form to calculate the equation for the line.

x1 = 2022

y1 = ₹9,00,000 cr

x2 = 2030

y2 = ₹35,00,000 cr

Using 2-point form the equation of line comes out to be:

x(t) = 3,25,000*t + 9,00,000

Substituting all the values, we get

y = 3,25,000*m*t + 9,00,000m – 16,90,0000

The pricing(m) can be adjusted according to the desired profits(y).

Conclusion:

Overall, the OCR technology market follows an upward trend and seems to be a promising investment. Cost-cutting can be effectively performed and scalability of the service is convenient.

Since all the factors are favourable, the competition is high and competitive pricing would go a long way.