Aim: To build a Cognitive based application to acquire knowledge through images for a Customer service application/ Insurance/ Healthcare Application/ Smarter Cities/Government etc.

Theory:

In this experiment, we develop a cognitive application that can understand and extract useful knowledge from images, especially real-world documents like electricity bills. This type of system falls under the field of Cognitive Computing, where the goal is to simulate human-like understanding through machines.

Here, the application is designed to help customers in smart cities or government utilities by analyzing their electricity bills and providing insights like:

- Total bill amount
- Units consumed
- Due date
- Suggestions to reduce usage

What Makes It Cognitive?

The system is called cognitive because it does more than just scan an image. It performs the following intelligent tasks:

1. Reads the bill image like a human would
2. Understands the content such as charges, usage, and dates
3. Analyzes the data for patterns (e.g., high usage)
4. Responds with useful suggestions or actions

Image-Based Knowledge Extraction

The core idea is to use images as input and turn them into structured knowledge using a combination of:

- Computer Vision: To understand and clean the image
- OCR (Optical Character Recognition): To convert image text into digital text
- Regex (Regular Expressions): To find and extract key data like bill amount and units

This allows the system to work directly with scanned bills, mobile photos, or printed reports, which are commonly used in customer service scenarios.

Tools & Techniques Used

Tool/Concept	Role in the System
OpenCV	Image preprocessing (grayscale, thresholding, noise removal)
pytesseract	Optical Character Recognition to extract text from image
Regex (re module)	Identify and extract meaningful values like total bill, units consumed, etc.
Python	Main language for programming the application logic

Workflow of the System

1. Image Upload: The user uploads a photo or scanned copy of the electricity bill.
2. Preprocessing: The image is cleaned using filters to improve clarity (grayscale, thresholding).
3. OCR Extraction: All readable text is extracted from the image using Tesseract OCR.
4. Data Extraction: The text is searched using patterns to identify important values.
5. Result & Feedback: The extracted information is shown to the user, along with smart suggestions like:
    - "Reduce air conditioner usage"
    - "Consider using energy-efficient lighting"
6. End Output: Helpful insights are displayed to assist the user in saving electricity and managing bills.

Real-Life Use Case in Smarter Cities

In a smart electricity usage system, this cognitive application could be integrated with:

- Government energy dashboards
- Customer self-service portals
- Utility billing and automation systems

This reduces human errors, speeds up query handling, and improves energy awareness among citizens.

```
# Install required libraries
!pip install pytesseract opencv-python pillow --quiet
```

```
import cv2
import pytesseract
import re
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
```

```
# Upload real electricity bill image
uploaded = files.upload()
```

Choose Files View-MSE…ashtra.jpg.jpg

- **View-MSEDCL-Bill-Maharashtra.jpg.jpg**(image/jpeg) - 107508 bytes, last modified: 8/30/2025 - 100% done

```
Saving View-MSEDCL-Bill-Maharashtra.jpg.jpg to View-MSEDCL-Bill-Maharashtra.jpg (1).jpg
```

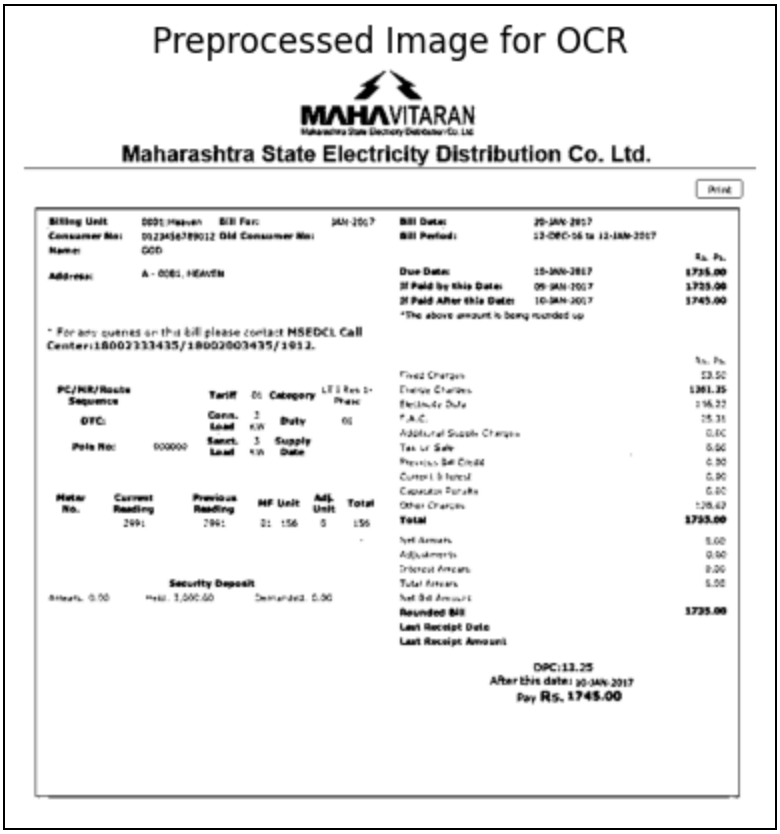
```
# Pick the uploaded file
for file_name in uploaded.keys():
    image_path = file_name
    break
```

```
# Load image using OpenCV
img = cv2.imread(image_path)
```

```
# Convert to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
# Apply threshold for better OCR
gray = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]
```

```
# Show preprocessed image
plt.imshow(gray, cmap='gray')
plt.axis('off')
plt.title("Preprocessed Image for OCR")
plt.show()
```




```
# OCR extraction
extracted_text = pytesseract.image_to_string(gray)
```

```
# Show full text (optional)
print("Extracted Text:\n", extracted_text)
```

```
Extracted Text:
 ex

MAHAVITARAN

anstteg Sa Bacooy Dabber Oo id

Maharashtra State Electricity Distribution Co. Ltd.

aan? roman?
Poee te reuav207
Duepae toaai2017
Nowa bythe Date on N-2017
Haid Aker foie Oates 1044-2017
'the seve attr n borg ded
MSEDCL Call
Centers38007323425/1800200299/2912.
PeynR/Rovte Wines Tae Crate
Fart 01 Cane
'Sequence recite ry Pave Bhcctnty Date
ore: fone 2 pay TN
Bra Suse Ch
potene: coorao Sane 3, Sunply een
Povo ba Cree
cine
```

```
eter current Previous Ak ton
Ne. Rasdiog ME Unit yeie Tota
2 a8

Security Deposit

aitean, 0.09 908.20

cemented. 6.06

Total areas
ot Bul Amur
Rounded eit

nt Receipt Date
lst Receipt Amount

9°C:13.25
'After this date: 10-24-2017
Pay RS. 1748.00

Ra, Pe
1738.00
1728.00
$745.00
```

```python
# Function to extract important details
def extract_bill_info(text):
    info = {}

    # Total Bill
    match_total = re.search(r'(?i)(Total|Rounded Bill|Pay
Rs\.?)[:\s]*₹?\s?(\d{3,6}\.?\d{0,2})', text)
    if match_total:
        info["Total Bill (₹)"] = match_total.group(2)

    # Due Date
    match_due = re.search(r'Due Date[:\s]+(\d{2}-[A-Z][a-z]{2}-\d{4})',
text)
    if match_due:
        info["Due Date"] = match_due.group(1)

    # Units Consumed
    match_units = re.search(r'(?i)(Units|Adj\.
Unit|Consumed)[:\s]+(\d{2,5})', text)
    if match_units:
        info["Units Consumed"] = match_units.group(2)

    return info
```

```
# Extract and print result
info = extract_bill_info(extracted_text)
print("\nFinal Extracted Information:")
for k, v in info.items():
    print(f"{k}: {v}")
```

```
Final Extracted Information:
Total Bill (₹): 1748.00
```

```
# Give suggestions
if info:
    print("\nSmart Suggestions:")
    if "Units Consumed" in info and int(info["Units Consumed"]) > 150:
        print("- High unit usage detected. Consider switching off unused
appliances.")
    if "Total Bill (₹)" in info and float(info["Total Bill (₹)"]) > 1500:
        print("- Your bill is high. Consider using LED lights or solar
solutions.")
else:
    print("Could not extract information. Try a clearer or higher-quality
image.")
```

```
Smart Suggestions:
- Your bill is high. Consider using LED lights or solar solutions.
```

Conclusion:

This experiment demonstrates how AI-powered cognitive systems can extract and interpret information from images, making them useful for real-world applications like bill analysis in smart cities and customer support. The approach makes daily tasks smarter, faster, and more helpful by combining vision, logic, and automation.

AI&DS2_Expt_03