eng21ds0002-abhishek-a-sisa-aiml

September 13, 2024

1 ASSIGNMENT (Problem Statement)

1.1 Python/ML Intern

1.1.1 Problem Statement

1. There are a huge number of files in a system. Some files may have data more than 1 GB.
2. Traverse through folders and files of a specific directory or folder. Read all files (text, document, PDF) and identify if any files contain credit card numbers.
   - Note: Create sample files with credit card numbers and test with at least 50 files; 20% should contain valid credit card numbers.
3. The algorithm should be optimized to scan files quickly while keeping CPU and memory consumption within 30%.

---

2 Solution

2.1 1. File Creation and Zipping

2.1.1 Import Required Libraries

[1]:
```
import os
import random
import string
import zipfile
```

2.1.2 Directory to Store Sample Files

[2]:
```
directory = 'test_files'
```

2.1.3 Sample Credit Card Patterns

[3]:
```
valid_credit_cards = [
    "4539 1488 0343 6467", "4716 6017 4402 1937", "4485 2357 2346 1636",
    "6011 1111 1111 1117", "6011 4432 3422 3456", "3782 822463 10005",
    "3714 496353 98431", "3056 930902 5904", "3852 000002 32323",
    "3530 111333 000000", "5555 5555 5555 4444"
]
```

2.1.4 Function to Generate Random Text Data

[4]:
```
def generate_random_text(size):
    return ''.join(random.choices(string.ascii_letters + string.digits + ' ',␣
 ↪k=size))
```

2.1.5 Function to Create Sample Files and Function to Zip the Directory

[5]:
```
def create_sample_files(directory, num_files=50, cc_percentage=20):
    if not os.path.exists(directory):
        os.makedirs(directory)

    num_files_with_cc = int(num_files * (cc_percentage / 100))
    for i in range(1, num_files + 1):
        filename = f"file_{i}.txt"
        filepath = os.path.join(directory, filename)

        # Decide whether to insert a credit card number in the file
        if i <= num_files_with_cc:
            # Insert credit card number with some random text
            content = generate_random_text(1000) + "\n" + random.
 ↪choice(valid_credit_cards) + "\n" + generate_random_text(1000)
        else:
            # Insert only random text
            content = generate_random_text(2000)

        # Write the content to the file
        with open(filepath, 'w') as f:
            f.write(content)

    print(f"{num_files} files created successfully in {directory}")

# Function to zip the directory
def zip_directory(directory, zip_file_path):
    with zipfile.ZipFile(zip_file_path, 'w') as zip_file:
        for foldername, subfolders, filenames in os.walk(directory):
            for filename in filenames:
                file_path = os.path.join(foldername, filename)
                zip_file.write(file_path, os.path.relpath(file_path, directory))

    print(f"Files zipped to {zip_file_path}")

# Create 50 test files with 20% containing credit card numbers
create_sample_files(directory)

# Zip the test_files directory
zip_directory(directory, 'test_files.zip')
```

```
50 files created successfully in test_files
Files zipped to test_files.zip
```

2.1.6 Link to download the sample generated text files

[6]:
```
from IPython.display import FileLink

# Display the link to download the zip file
FileLink('test_files.zip')
```

[6]:
```
C:\Users\91709\test_files.zip
```

2.2 2. Credit Card Detection

2.2.1 Import Additional Libraries

[7]:
```
import re
from concurrent.futures import ThreadPoolExecutor
from docx import Document
import pdfplumber
```

2.2.2 Credit Card Regex Pattern

[8]:
```
CREDIT_CARD_REGEX = r'\b(?:\d[ -]*?){13,16}\b'
```

2.2.3 Function to Traverse Directory and Get Files

[9]:
```
def get_text_files(directory):
    for root, dirs, files in os.walk(directory):
        for file in files:
            if file.endswith(('.txt', '.docx', '.pdf')):
                yield os.path.join(root, file)
```

2.2.4 Function to Detect Credit Card in File Content

[10]:
```
def detect_credit_card(content):
    return bool(re.search(CREDIT_CARD_REGEX, content))
```

2.2.5 Functions to Extract Text from .docx and .pdf Files

[11]:
```
def extract_text_from_docx(filepath):
    doc = Document(filepath)
    text = []
    for para in doc.paragraphs:
        text.append(para.text)
    return "\n".join(text)
```

```
def extract_text_from_pdf(filepath):
    text = []
    with pdfplumber.open(filepath) as pdf:
        for page in pdf.pages:
            text.append(page.extract_text())
    return "\n".join(text)
```

2.3 3. File Reading and Processing

2.3.1 To avoid high memory usage, read large files in chunks. Here's a way to efficiently scan through the files:

[12]:
```
def read_file(filepath, chunk_size=1024):
    ext = os.path.splitext(filepath)[1].lower()
    if ext == '.txt':
        with open(filepath, 'r', encoding='utf-8', errors='ignore') as file:
            while True:
                chunk = file.read(chunk_size)
                if not chunk:
                    break
                yield chunk
    elif ext == '.docx':
        content = extract_text_from_docx(filepath)
        for chunk in (content[i:i + chunk_size] for i in range(0, len(content),␣
↪chunk_size)):
            yield chunk
    elif ext == '.pdf':
        content = extract_text_from_pdf(filepath)
        for chunk in (content[i:i + chunk_size] for i in range(0, len(content),␣
↪chunk_size)):
            yield chunk
    else:
        raise ValueError(f"Unsupported file type: {ext}")
```

2.4 4. Optimized File Scanning with Thread Pool

2.4.1 Use Python's concurrent.futures.ThreadPoolExecutor to process multiple files in parallel. This helps distribute the CPU load efficiently

[13]:
```
def scan_file_for_credit_card(filepath, files_with_cc):
    print(f"Scanning {filepath}...")
    try:
        for chunk in read_file(filepath):
            if detect_credit_card(chunk):
                print(f"Credit card number found in {filepath}")
                files_with_cc.append(filepath)
```

```
                return
    except ValueError as e:
        print(e)


def scan_files_concurrently(directory, max_workers=5):
    files_with_cc = []
    with ThreadPoolExecutor(max_workers=max_workers) as executor:
        for filepath in get_text_files(directory):
            executor.submit(scan_file_for_credit_card, filepath, files_with_cc)
    return files_with_cc
```

2.5 5. Main Execution Block

[14]:
```
if __name__ == "__main__":
    target_directory = 'test_files'  # Adjust to your directory path
    files_with_credit_cards = scan_files_concurrently(target_directory)
```

```
Scanning test_files\file_1.txt…Scanning test_files\file_10.txt…

Scanning test_files\file_11.txt…
Credit card number found in test_files\file_10.txt
Credit card number found in test_files\file_1.txt
Scanning test_files\file_12.txt…
Scanning test_files\file_13.txt…
Scanning test_files\file_14.txt…
Scanning test_files\file_15.txt…
Scanning test_files\file_16.txt…
Scanning test_files\file_17.txt…
Scanning test_files\file_18.txt…
Scanning test_files\file_19.txt…
Scanning test_files\file_2.txt…
Scanning test_files\file_20.txt…
Scanning test_files\file_21.txt…
Scanning test_files\file_22.txt…
Scanning test_files\file_23.txt…
Credit card number found in test_files\file_2.txt
Scanning test_files\file_24.txt…
Scanning test_files\file_25.txt…
Scanning test_files\file_26.txt…
Scanning test_files\file_27.txt…
Scanning test_files\file_28.txt…
Scanning test_files\file_29.txt…
Scanning test_files\file_3.txt…
Scanning test_files\file_30.txt…
Scanning test_files\file_31.txt…
Scanning test_files\file_32.txt…
```

```
Scanning test_files\file_33.txt…
Credit card number found in test_files\file_3.txt
Scanning test_files\file_34.txt…
Scanning test_files\file_35.txt…
Scanning test_files\file_36.txt…
Scanning test_files\file_37.txt…
Scanning test_files\file_38.txt…
Scanning test_files\file_39.txt…
Scanning test_files\file_4.txt…
Scanning test_files\file_40.txt…
Scanning test_files\file_41.txt…
Credit card number found in test_files\file_4.txt
Scanning test_files\file_42.txt…
Scanning test_files\file_43.txt…
Scanning test_files\file_44.txt…
Scanning test_files\file_45.txt…
Scanning test_files\file_46.txt…
Scanning test_files\file_47.txt…
Scanning test_files\file_48.txt…
Scanning test_files\file_49.txt…
Scanning test_files\file_5.txt…
Scanning test_files\file_50.txt…
Credit card number found in test_files\file_5.txt
Scanning test_files\file_6.txt…
Scanning test_files\file_7.txt…
Scanning test_files\file_8.txt…
Scanning test_files\file_9.txt…
Credit card number found in test_files\file_8.txt
Credit card number found in test_files\file_6.txt
Credit card number found in test_files\file_7.txt
Credit card number found in test_files\file_9.txt
```

[15]:
```python
# Print out the list of files that contain credit card numbers
print("\nFiles with credit card numbers detected:")
for file in files_with_credit_cards:
        print(file)
```

```
Files with credit card numbers detected:
test_files\file_10.txt
test_files\file_1.txt
test_files\file_2.txt
test_files\file_3.txt
test_files\file_4.txt
test_files\file_5.txt
test_files\file_8.txt
test_files\file_6.txt
test_files\file_7.txt
```

```
test_files\file_9.txt
```

2.6 6. Putting It All Together

2.6.1 Here's how the full script would look:

[16]:
```python
import os
import random
import string
import zipfile

directory = 'test_files'

valid_credit_cards = [
    "4539 1488 0343 6467", "4716 6017 4402 1937", "4485 2357 2346 1636",
    "6011 1111 1111 1117", "6011 4432 3422 3456", "3782 822463 10005",
    "3714 496353 98431", "3056 930902 5904", "3852 000002 32323",
    "3530 111333 000000", "5555 5555 5555 4444"
]

def generate_random_text(size):
    return ''.join(random.choices(string.ascii_letters + string.digits + ' ',␣
 ↪k=size))

def create_sample_files(directory, num_files=50, cc_percentage=20):
    if not os.path.exists(directory):
        os.makedirs(directory)

    num_files_with_cc = int(num_files * (cc_percentage / 100))
    for i in range(1, num_files + 1):
        filename = f"file_{i}.txt"
        filepath = os.path.join(directory, filename)

        # Decide whether to insert a credit card number in the file
        if i <= num_files_with_cc:
            # Insert credit card number with some random text
            content = generate_random_text(1000) + "\n" + random.
 ↪choice(valid_credit_cards) + "\n" + generate_random_text(1000)
        else:
            # Insert only random text
            content = generate_random_text(2000)

        # Write the content to the file
        with open(filepath, 'w') as f:
            f.write(content)

    print(f"{num_files} files created successfully in {directory}")
```

```
# Function to zip the directory
def zip_directory(directory, zip_file_path):
    with zipfile.ZipFile(zip_file_path, 'w') as zip_file:
        for foldername, subfolders, filenames in os.walk(directory):
            for filename in filenames:
                file_path = os.path.join(foldername, filename)
                zip_file.write(file_path, os.path.relpath(file_path, directory))

    print(f"Files zipped to {zip_file_path}")

# Create 50 test files with 20% containing credit card numbers
create_sample_files(directory)

# Zip the test_files directory
zip_directory(directory, 'test_files.zip')


from IPython.display import FileLink

# Display the link to download the zip file
FileLink('test_files.zip')


import re
from concurrent.futures import ThreadPoolExecutor
from docx import Document
import pdfplumber

CREDIT_CARD_REGEX = r'\b(?:\d[ -]*?){13,16}\b'

def get_text_files(directory):
    for root, dirs, files in os.walk(directory):
        for file in files:
            if file.endswith(('.txt', '.docx', '.pdf')):
                yield os.path.join(root, file)



def detect_credit_card(content):
    return bool(re.search(CREDIT_CARD_REGEX, content))


def extract_text_from_docx(filepath):
    doc = Document(filepath)
    text = []
    for para in doc.paragraphs:
        text.append(para.text)
```

```
    return "\n".join(text)

def extract_text_from_pdf(filepath):
    text = []
    with pdfplumber.open(filepath) as pdf:
        for page in pdf.pages:
            text.append(page.extract_text())
    return "\n".join(text)


def read_file(filepath, chunk_size=1024):
    ext = os.path.splitext(filepath)[1].lower()
    if ext == '.txt':
        with open(filepath, 'r', encoding='utf-8', errors='ignore') as file:
            while True:
                chunk = file.read(chunk_size)
                if not chunk:
                    break
                yield chunk
    elif ext == '.docx':
        content = extract_text_from_docx(filepath)
        for chunk in (content[i:i + chunk_size] for i in range(0, len(content),␣
 ↪chunk_size)):
            yield chunk
    elif ext == '.pdf':
        content = extract_text_from_pdf(filepath)
        for chunk in (content[i:i + chunk_size] for i in range(0, len(content),␣
 ↪chunk_size)):
            yield chunk
    else:
        raise ValueError(f"Unsupported file type: {ext}")


def scan_file_for_credit_card(filepath, files_with_cc):
    print(f"Scanning {filepath}...")
    try:
        for chunk in read_file(filepath):
            if detect_credit_card(chunk):
                print(f"Credit card number found in {filepath}")
                files_with_cc.append(filepath)
                return
    except ValueError as e:
        print(e)


def scan_files_concurrently(directory, max_workers=5):
    files_with_cc = []
```

```
    with ThreadPoolExecutor(max_workers=max_workers) as executor:
        for filepath in get_text_files(directory):
            executor.submit(scan_file_for_credit_card, filepath, files_with_cc)
    return files_with_cc


if __name__ == "__main__":
    target_directory = 'test_files'  # Adjust to your directory path
    files_with_credit_cards = scan_files_concurrently(target_directory)

# Print out the list of files that contain credit card numbers
print("\nFiles with credit card numbers detected:")
for file in files_with_credit_cards:
        print(file)
```

```
50 files created successfully in test_files
Files zipped to test_files.zip
Scanning test_files\file_1.txt…
Credit card number found in test_files\file_1.txt
Scanning test_files\file_10.txt…
Credit card number found in test_files\file_10.txt
Scanning test_files\file_11.txt…
Scanning test_files\file_12.txt…
Scanning test_files\file_13.txt…
Scanning test_files\file_14.txt…
Scanning test_files\file_15.txt…
Scanning test_files\file_16.txt…
Scanning test_files\file_17.txt…
Scanning test_files\file_18.txt…
Scanning test_files\file_19.txt…
Scanning test_files\file_2.txt…
Scanning test_files\file_20.txt…
Scanning test_files\file_21.txt…
Scanning test_files\file_22.txt…
Credit card number found in test_files\file_2.txt
Scanning test_files\file_23.txt…
Scanning test_files\file_24.txt…
Scanning test_files\file_25.txt…
Scanning test_files\file_26.txt…
Scanning test_files\file_27.txt…
Scanning test_files\file_28.txt…
Scanning test_files\file_29.txt…
Scanning test_files\file_3.txt…
Scanning test_files\file_30.txt…
Credit card number found in test_files\file_3.txt
Scanning test_files\file_31.txt…
Scanning test_files\file_32.txt…
```

```
Scanning test_files\file_33.txt…
Scanning test_files\file_34.txt…
Scanning test_files\file_35.txt…
Scanning test_files\file_36.txt…
Scanning test_files\file_37.txt…
Scanning test_files\file_38.txt…
Scanning test_files\file_39.txt…
Scanning test_files\file_4.txt…
Scanning test_files\file_40.txt…
Scanning test_files\file_41.txt…
Scanning test_files\file_42.txt…
Credit card number found in test_files\file_4.txt
Scanning test_files\file_43.txt…
Scanning test_files\file_44.txt…
Scanning test_files\file_45.txt…
Scanning test_files\file_46.txt…
Scanning test_files\file_47.txt…
Scanning test_files\file_48.txt…
Scanning test_files\file_49.txt…
Scanning test_files\file_5.txt…
Scanning test_files\file_50.txt…
Scanning test_files\file_6.txt…
Credit card number found in test_files\file_5.txt
Scanning test_files\file_7.txt…
Scanning test_files\file_8.txt…
Scanning test_files\file_9.txt…
Credit card number found in test_files\file_6.txt
Credit card number found in test_files\file_7.txt
Credit card number found in test_files\file_8.txt
Credit card number found in test_files\file_9.txt

Files with credit card numbers detected:
test_files\file_1.txt
test_files\file_10.txt
test_files\file_2.txt
test_files\file_3.txt
test_files\file_4.txt
test_files\file_5.txt
test_files\file_6.txt
test_files\file_7.txt
test_files\file_8.txt
test_files\file_9.txt
```

[ ]: