import os

import re

from pathlib import Path

import pdfplumber

from docx import Document

def sanitize\_text(text):

"""

Sanitize text by replacing sensitive information with placeholders.

"""

phone\_pattern = r"(\+1[-.\s]?\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4}|\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4})"

email\_pattern = r"([a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA0-9]{2,})"

college\_names\_pattern = r"\b(?:University|College|Institute|School)\b[^\.]\*\b\w+\b"

text = re.sub(phone\_pattern, lambda match: "[REDACTED\_PHONE]", text)

text = re.sub(email\_pattern, lambda match: "[REDACTED\_EMAIL]", text)

def replace\_college\_once(match):

if not hasattr(replace\_college\_once, "replaced"):

replace\_college\_once.replaced = True

return "[REDACTED\_COLLEGE]"

return match.group()

text = re.sub(college\_names\_pattern, replace\_college\_once, text, flags=re.IGNORECASE)

return text

def process\_pdf(file\_path, output\_folder):

"""

Process PDF, sanitize content, and replicate professional formatting in a Word document.

"""

print(f"Processing PDF: {file\_path}")

with pdfplumber.open(file\_path) as pdf:

sanitized\_doc = Document()

for page\_num, page in enumerate(pdf.pages):

text = page.extract\_text()

if text:

sanitized\_text = sanitize\_text(text)

# Split into paragraphs based on newlines and list markers

paragraphs = re.split(r"\n\n|\n(?=\S)|\n\s\*[-•]", sanitized\_text)

for paragraph in paragraphs:

# Add headings and sub-headings with consistent styling

if paragraph.lower().startswith("education") or "degree" in paragraph.lower():

sanitized\_doc.add\_heading(paragraph.strip(), level=1)

elif paragraph.lower().startswith("skills") or "language" in paragraph.lower():

sanitized\_doc.add\_heading(paragraph.strip(), level=2)

elif "summary" in paragraph.lower():

sanitized\_doc.add\_heading(paragraph.strip(), level=2)

elif paragraph.lower().startswith("experience") or "research" in paragraph.lower():

sanitized\_doc.add\_heading(paragraph.strip(), level=2)

elif paragraph.startswith("### "):

sanitized\_doc.add\_heading(paragraph[4:], level=3)

elif paragraph.startswith("## "):

sanitized\_doc.add\_heading(paragraph[3:], level=2)

elif paragraph.startswith("# "):

sanitized\_doc.add\_heading(paragraph[2:], level=1)

else:

sanitized\_doc.add\_paragraph(paragraph.strip())

sanitized\_doc.add\_paragraph() # Add space between pages

output\_path = os.path.join(output\_folder, Path(file\_path).stem + "\_sanitized.docx")

sanitized\_doc.save(output\_path)

print(f"Sanitized Word document saved at: {output\_path}")

def process\_docx(file\_path, output\_folder):

"""

Sanitize Word document while preserving original formatting, including inline styles and tables.

"""

print(f"Processing Word document: {file\_path}")

doc = Document(file\_path)

sanitized\_doc = Document()

# Copy all paragraphs with inline formatting

for paragraph in doc.paragraphs:

sanitized\_paragraph = sanitized\_doc.add\_paragraph()

sanitized\_paragraph.style = paragraph.style

sanitized\_paragraph.alignment = paragraph.alignment

for run in paragraph.runs:

sanitized\_run = sanitized\_paragraph.add\_run()

sanitized\_run.bold = run.bold

sanitized\_run.italic = run.italic

sanitized\_run.underline = run.underline

sanitized\_run.font.size = run.font.size

sanitized\_run.font.name = run.font.name

sanitized\_run.text = sanitize\_text(run.text)

# Copy and sanitize tables

for table in doc.tables:

sanitized\_table = sanitized\_doc.add\_table(rows=len(table.rows), cols=len(table.columns))

sanitized\_table.style = table.style

for i, row in enumerate(table.rows):

for j, cell in enumerate(row.cells):

sanitized\_cell = sanitized\_table.cell(i, j)

sanitized\_cell.text = sanitize\_text(cell.text)

sanitized\_cell.paragraphs.style = cell.paragraphs.style

# Save the sanitized document

output\_path = os.path.join(output\_folder, Path(file\_path).stem + "\_sanitized.docx")

sanitized\_doc.save(output\_path)

print(f"Sanitized Word document saved at: {output\_path}")

def sanitize\_resumes(input\_path, output\_folder):

"""

Sanitize resumes from a file or directory while preserving formatting.

"""

os.makedirs(output\_folder, exist\_ok=True)

print(f"Output folder: {output\_folder}")

if os.path.isfile(input\_path):

file\_name = os.path.basename(input\_path)

if file\_name.endswith(".pdf"):

process\_pdf(input\_path, output\_folder)

elif file\_name.endswith(".docx"):

process\_docx(input\_path, output\_folder)

else:

print(f"Unsupported file type: {file\_name}")

elif os.path.isdir(input\_path):

print(f"Processing directory: {input\_path}")

for file\_name in os.listdir(input\_path):

file\_path = os.path.join(input\_path, file\_name)

if file\_name.endswith(".pdf"):

process\_pdf(file\_path, output\_folder)

elif file\_name.endswith(".docx"):

process\_docx(file\_path, output\_folder)

else:

print(f"Skipping unsupported file: {file\_name}")

else:

print(f"Invalid path: {input\_path}")

# Example usage

input\_path = r"D:\AICTE\AIandMLIntern\_20241127\abdollahi\_armin\_resume\_51801848.pdf"

output\_folder = r"D:\AICTE\AIandMLIntern\_20241127\output"

sanitize\_resumes(input\_path, output\_folder)