Mise à l’échelle du contenu PDF pour l’impression

J’ai besoin d’imprimer des documents, et j’ai remarqué que l’espace blanc autour du contenu est souvent trop grand, ce qui gaspille du papier et rend le texte plus petit qu’il ne devrait l’être. Ce script aide à redimensionner automatiquement le contenu du PDF pour mieux l’adapter à la page en détectant la zone de contenu et en la redimensionnant pour remplir la page, tout en respectant une petite marge.

```
import subprocess
import sys
import os
from PIL import Image
from pdf2image import convert_from_path

MARGIN_PERCENT = 0.005
DPI = 72

def convert_pixels_to_points(pixels, dpi):
    """Convertit les pixels en points."""
    return pixels * 72 / dpi

def get_image_dimensions(image):
    """Obtient les dimensions de l'image en pixels et en points."""
    width, height = image.size
    dpi = image.info.get('dpi', (DPI, DPI))
    width_points = convert_pixels_to_points(width, dpi[0])
    height_points = convert_pixels_to_points(height, dpi[1])
    return width, height, width_points, height_points, dpi

def analyze_whitespace(image, width, height):
    """Analyse l'espace blanc pour trouver la boîte englobante du contenu."""
    left_margin_px = width
    right_margin_px = 0
    top_margin_px = height
    bottom_margin_px = 0
    found_content = False

    for x in range(width):
        for y in range(height):
            pixel = image.getpixel((x, y))
            if isinstance(pixel, tuple):
```

```
                if any(c < 250 for c in pixel):
                    if not found_content:
                        left_margin_px = x
                        top_margin_px = y
                        found_content = True
                    right_margin_px = max(right_margin_px, x)
                    bottom_margin_px = max(bottom_margin_px, y)
            elif pixel < 250:
                if not found_content:
                    left_margin_px = x
                    top_margin_px = y
                    found_content = True
                right_margin_px = max(right_margin_px, x)
                bottom_margin_px = max(bottom_margin_px, y)

    if not found_content:
        return None, None, None, None

    right_margin_px = width - right_margin_px
    bottom_margin_px = height - bottom_margin_px
    return left_margin_px, right_margin_px, top_margin_px, bottom_margin_px

def calculate_scale_factor(input_pdf):
    """
    Détecte les dimensions de la première page d'un PDF, analyse l'espace blanc,
    et calcule le facteur d'échelle en fonction du contenu du PDF et des dimensions cibles A4 avec marges.
    Retourne le facteur d'échelle ou None si une erreur survient.
    """
    print(f"Calcul du facteur d'échelle pour : {input_pdf}")
    try:
        images = convert_from_path(input_pdf, first_page=1, last_page=1)
        if not images:
            print("  Impossible de convertir le PDF en image.")
            return None

        image = images[0]
        width, height, width_points, height_points, dpi = get_image_dimensions(image)

        margins = analyze_whitespace(image, width, height)
        if margins[0] is None:
```

```
        print("  Impossible de déterminer la boîte englobante du contenu.")
        left_margin_points = 0
        right_margin_points = 0
        top_margin_points = 0
        bottom_margin_points = 0
    else:
        left_margin_px, right_margin_px, top_margin_px, bottom_margin_px = margins
        content_width_px = right_margin_px - left_margin_px
        content_height_px = bottom_margin_px - top_margin_px

        left_margin_points = convert_pixels_to_points(left_margin_px, dpi[0])
        right_margin_points = convert_pixels_to_points(right_margin_px, dpi[0])
        top_margin_points = convert_pixels_to_points(top_margin_px, dpi[1])
        bottom_margin_points = convert_pixels_to_points(bottom_margin_px, dpi[1])

        print(f"  Boîte de contenu : gauche={left_margin_px}, haut={top_margin_px}, droite={right_margin_p
        print(f"  Dimensions du contenu (pixels) : largeur={content_width_px}, hauteur={content_height_px
        print(f"  Marges (points) : gauche={left_margin_points}, droite={right_margin_points}, haut={top_m

    print(f"  Dimensions détectées : largeur={width_points}, hauteur={height_points}")

    width_margin_points = min(left_margin_points, right_margin_points)
    height_margin_points = min(top_margin_points, bottom_margin_points)

    content_width = width_points - width_margin_points * 2
    content_height = height_points - height_margin_points * 2

    target_width = width_points * (1 - 2 * MARGIN_PERCENT)
    target_height = height_points * (1- 2 * MARGIN_PERCENT)

    width_scale = target_width / content_width
    height_scale = target_height / content_height

    print(f"  Dimensions du contenu (points) : largeur={content_width}, hauteur={content_height}")

    if content_width <= 0 or content_height <= 0:
        print("Erreur : Impossible de déterminer les dimensions du contenu.")
        return None

    print(f"  Dimensions cibles : largeur={target_width}, hauteur={target_height}")
```

```
        print(f"  Facteur d'échelle calculé en largeur : {width_scale}, en hauteur : {height_scale}")

        scale_factor = min(width_scale, height_scale)
        print(f"  Facteur d'échelle final : {scale_factor}")

        return scale_factor

    except Exception as e:
        print(f"Erreur lors de l'obtention des dimensions du PDF ou du calcul du facteur d'échelle : {e}")
        return None


def scale_pdf(input_pdf, output_pdf, scale_factor):
    """Redimensionne un PDF en utilisant pdfjam."""
    print(f"Redimensionnement de {input_pdf} vers {output_pdf} avec un facteur d'échelle : {scale_factor}")
    try:
        subprocess.run(
            [
                "pdfjam",
                "--scale",
                str(scale_factor),
                input_pdf,
                "--outfile",
                output_pdf,
            ],
            check=True,
        )
        print(f"Redimensionnement réussi de {input_pdf} vers {output_pdf}")
    except subprocess.CalledProcessError as e:
        print(f"Erreur lors du redimensionnement du PDF : {e}")
    except FileNotFoundError:
        print("Erreur : Commande pdfjam introuvable. Veuillez vous assurer qu'elle est installée et dans le PA


if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Utilisation : python scale-pdf.py <input_pdf> <output_pdf>")
        sys.exit(1)

    input_pdf = sys.argv[1]
```

```
output_pdf = sys.argv[2]
print(f"PDF d'entrée : {input_pdf}, PDF de sortie : {output_pdf}")

if not os.path.exists(input_pdf):
    print(f"Erreur : Fichier PDF d'entrée introuvable : {input_pdf}")
    sys.exit(1)

scale_factor = calculate_scale_factor(input_pdf)
if scale_factor is None:
    sys.exit(1)

scale_pdf(input_pdf, output_pdf, scale_factor)
```