Extracția și Preprocesarea Logo-urilor

1 Introducere

Logourile sunt esențiale pentru identitatea vizuală a unei companii. Acest document prezintă abordarea utilizată pentru a extrage automat logourile de pe site-uri web, eliminând fundalurile albe sau colorate și optimizând imaginile rezultate.

2 Pașii abordării

2.1 1. Extracția logo-ului din meta tag-uri

Dacă un site web conține meta tag-uri precum 'og:image' sau 'twitter:image', acestea sunt cele mai bune surse pentru identificarea logo-ului. Codul utilizează BeautifulSoup pentru a parcurge structura HTML și a extrage URL-ul imaginii.

2.2 2. Identificarea logo-ului în HTML

Dacă nu există meta tag-uri relevante, se caută logo-ul în elementele '¡img¿' cu clase sau ID-uri relevante ('logo', 'header-logo', 'site-logo'). Se utilizează XPath pentru identificare.

2.3 3. Descărcarea și procesarea imaginilor

Imaginile extrase sunt descărcate și salvate local folosind biblioteca 'requests'.

2.4 4. Eliminarea fundalului folosind rembg

'rembg' este o bibliotecă Python bazată pe modele de deep learning care elimină fundalul unei imagini.

```
from rembg import remove

def remove_background(image_path):
    with open(image_path, "rb") as img_file:
        input_data = img_file.read()
    output_data = remove(input_data)
    with open(image_path, "wb") as out_file:
```

```
        out_file.write(output_data)
```

2.5 5. Eliminarea fundalului colorat folosind OpenCV

Pentru a elimina fundalurile colorate, se utilizează OpenCV pentru a identifica culoarea predominantă de pe margini și a genera o mască pentru eliminarea acesteia.

```
import cv2
import numpy as np

def remove_colored_background(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    h, w, _ = image.shape
    edge_pixels = np.vstack([
        hsv[0, :, :], hsv[h-1, :, :],
        hsv[:, 0, :], hsv[:, w-1, :]
    ])
    dominant_color = np.median(edge_pixels, axis=0)
    h_val, s_val, v_val = dominant_color

    lower_bound = np.array([h_val - 10, max(50, s_val - 50), max(50, v_val - 50)
    upper_bound = np.array([h_val + 10, min(255, s_val + 50), min(255, v_val + 5
    mask = cv2.inRange(hsv, lower_bound, upper_bound)
    mask_inv = cv2.bitwise_not(mask)
    alpha = np.zeros_like(mask, dtype=np.uint8)
    alpha[mask_inv > 0] = 255
    bg_removed = cv2.merge((image[:, :, 0], image[:, :, 1], image[:, :, 2], alph
    cv2.imwrite(image_path, bg_removed)
```

2.6 6. Optimizarea performanței cu multiprocessing

Pentru a procesa multiple site-uri simultan, utilizăm 'multiprocessing.Pool', permițând rularea mai multor procese în paralel.

```
import multiprocessing

def process_logos_parallel(websites):
    with multiprocessing.Pool(processes=4) as pool:
        pool.map(extract_logo, websites)
```

3 Concluzie

Această metodă combină tehnici avansate de extracție a logo-urilor, eliminare a fundalului alb/colorat și optimizare a vitezei pentru un proces robust și scalabil. Extensii viitoare pot include detectarea automată a formatului transparent și compresia inteligentă a imaginilor.