Tłumacz japońskiego

Tłumaczenie połączenia znaków systemu KATAKANA na język angielski.

Wykonali:

Hańderek Natalia,

Huczek Krzysztof,

Jarosz Przemysław,

Kot Aleksandra

Spis treści

1. Wstęp

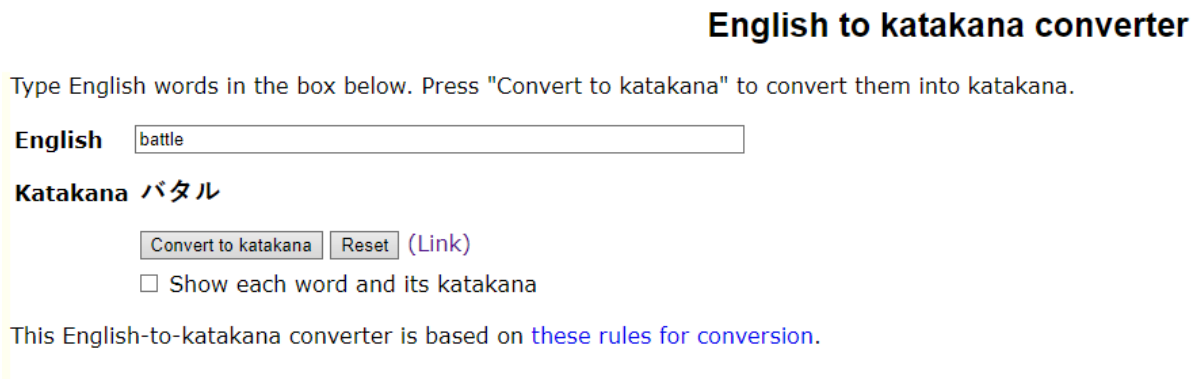
1.1 Cel projektu

Celem projektu było napisanie programu, który zapewniał tłumaczenie połączenia znaków systemu KATAKANA na język angielski.

1.2 Katakana

Katakana to jeden z dwóch japońskich systemów pisma sylabicznego kana. Drugi to hiragana.

1.3 Wykorzystany interpreter



Rysunek 1 źródło: https://www.sljfaq.org/cgi/e2k.cgi?word=battle

2. Implementacja

– W celu poprawnego wykonania zadania, potrzebny był import kilku komponentów:

```
import numpy as np
from skimage.io import imread, imshow
import matplotlib.pyplot as plt
from matplotlib.patches import Circle, Rectangle
from skimage import transform
from skimage.color import rgb2gray
from skimage.feature import match_template
from skimage.feature import peak_local_max
import matplotlib.font_manager as fm
from PIL import Image, ImageDraw, ImageFont
from difflib import get_close_matches
from os import listdir
from os.path import isfile, join
```

Skimage – do odczytu znaków z obrazków.

Matplotlib – do wyświetlania wykresów.

Difflib – do wyszukiwania najlepszego pasującego tłumaczenia ze słownika.

Os – do wylistowania plików zawierających zdjęcia znaków znajdujących się w folderze.

– Przygotowanie czcionki obsługującej japońskie znaki:

```
#font_path = '/usr/share/fonts/truetype/takao-gothic/TakaoGothic.ttf'
font_path = 'C:\Windows\Fonts\msgothic.ttc'
font_size = 128
font_color = (0,0,0)
font_background_color = 'white'
font = ImageFont.truetype(font_path, size=font_size)
font_properties = fm.FontProperties(fname=font_path)
```

– Metody służące do wygenerowania obrazków, które zawierają pojedyncze litery, a następnie ich scalenie w całe słowa:

```
# Generacja obrazkow
def generate_letter_images(letters):
    for letter in letters:
        img = Image.new('RGB', (font_size, font_size), color=font_background_color)
        imgDraw = ImageDraw.Draw(img)
        imgDraw.text((0, 0), letter, fill=font_color, font=font)
        img.save(f'letters_jpn\\auto\\{letter}.jpg', quality=120, format="JPEG")

def generate_words_images(dictionary):
    for jpn_word,eng_word in dictionary.items():
        jp_word = " ".join(list(jpn_word))
        img = Image.new('RGB', ((len(jp_word)) * font_size, font_size + 100), color=font_background_color)
        imgDraw = ImageDraw.Draw(img)
        imgDraw.text((100, 50), jp_word, fill=font_color, font=font)
        img.save(f'words_jpn\\auto\\{eng_word}.jpg', quality=120, format="JPEG")
```

– Metoda do pobierania obrazków, a następnie konwertowania ich na biało-czarne:

```
# Ladowanie obrazkow
def load_image(path):
    img = imread(path)
    return rgb2gray(img)
```

– Metoda do pobierania unikalnych liter, które zawierają słowa japońskie wprowadzone do słownika:

```
def get_letters(keys):
    letters = set()
    for key in keys:
        for letter in key:
            letters.add(letter)
    return list(letters)
```

– Metoda do wylistowania plików, które znajdują się w danym folderze. Potrzebna do przejścia pętlą po wygenerowanych obrazkach zawierających słowa:

```
def get_files(path):
    return [f for f in listdir(path) if isfile(join(path, f))]
```

– Funkcja jako argument pobiera obrazek ze słowem w języku japońskim, oraz listę obrazków z poszczególnymi literami. Dla każdej z liter sprawdza, czy znajduje się ona na obrazku. Jeśli się znajduje, zapisuje jej współrzędną x. Po przesortowaniu liter względem tej współrzędnej możemy określić w jakiej kolejności litery wystąpiły na obrazku, tj. jakie słowo utworzyły. Najważniejsze tutaj są metody match_template oraz peak_local_max z biblioteki skimage.feature:

```
def find_templates_in_image(img, letter_templates):
    letters_with_x_coord_pairs = []

    for i,letter_image in enumerate(letter_templates):
        resulting_image = match_template(img, letter_image)
        peaks = peak_local_max(resulting_image, threshold_abs=0.9, min_distance=10)

        if is_letter_found_on_image(peaks):
            plt.figure(num=None, figsize=(6,4))
            plot_resulting_image(resulting_image)
            plot_word_image(img)

            for y,x in peaks:
                if found_letter_should_be_considered(letters_with_x_coord_pairs, x):
                    letters_with_x_coord_pairs.append((letters[i], x))
                    draw_rectangle_around_letter((x,y), letter_image, letters[i])
                    draw_letter_above_rectangle((x,y), letters[i])

        plt.show()

    return letters_with_x_coord_pairs
```

match_template – funkcja zwraca obrazek tego typu:

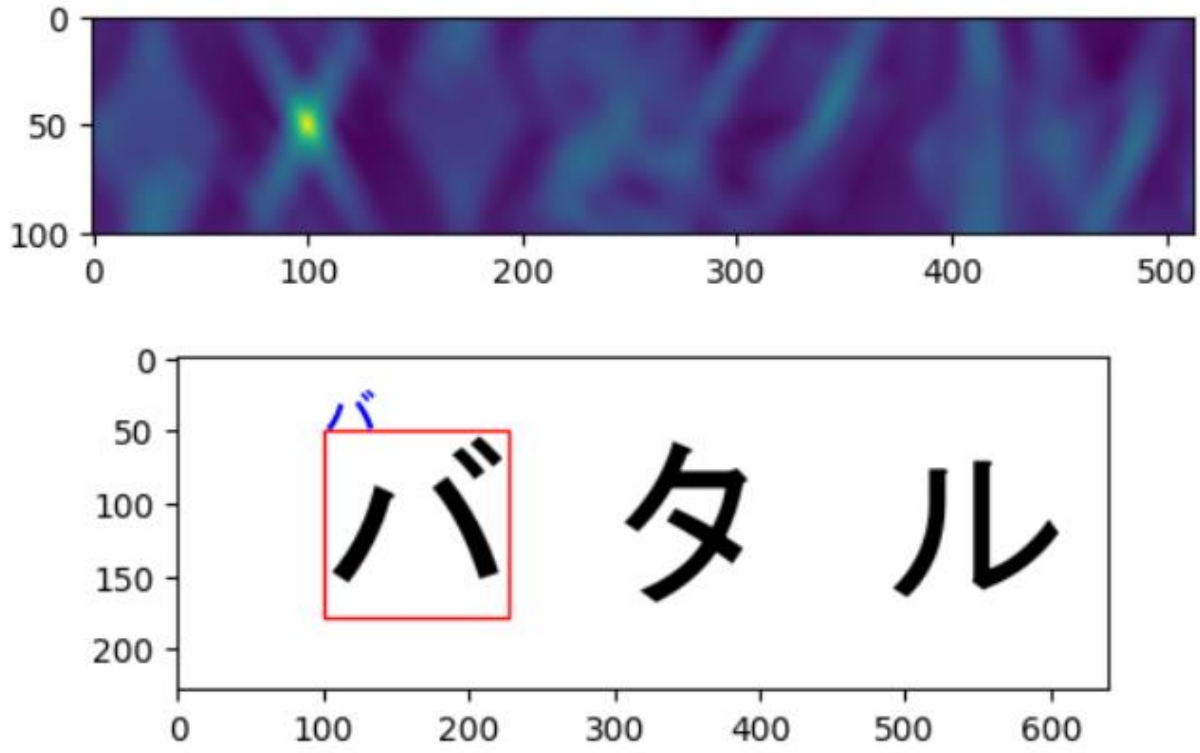
W obszarze najjaśniejszym jest największe prawdopodobieństwo, że znaleźliśmy fragment (literę), którego szukamy. Obszar ten jest lokalnym maksimum, który jesteśmy w stanie wyciągnąć za pomocą funkcji peak_local_max.

– Pobranie nazw plików, a następnie uruchomienie tłumaczeń dla każdego z wyrazów:

```
files = get_files('words_jpn/auto')

for file in files:
    img = load_image(f'words_jpn/auto/' + file)
    letters_with_x_coord_pairs = find_templates_in_image(img, letter_images)
    display_results(letters_with_x_coord_pairs, dictionary)
```

– Program przechodzi przez wszystkie dostępne znaki i określa, czy znajdują się one na obrazku. Znaleziony znak zostaje zakreślony czerwoną ramką:



Po przejściu wszystkich znaków, wyświetla raport odnośnie tego, co udało mu się znaleźć. Następnie tłumaczy znaleziony wyraz na jego odpowiednik w języku angielskim. Jeśli nie znajdzie dokładnie tego samego wyrazu, wyświetla ten który jest jemu ‚najbliższy'.

```
Found letter バ on x 100.
Found letter ル on x 484.
Found letter タ on x 292.
The letters in order formed a word: バタル
Best matching word available in dict: ('バタル', 'battle')
```

3. Podsumowanie

Cel pracy został osiągnięty.