Opis Projektu

Utworzony program pełni funkcję inteligentnej bazy danych osobowych, która gromadzi dane takie jak imię i nazwisko oraz twarz osoby. Po przesłaniu dokumentu lub zdjęcia program gromadzi dane przedstawione na obrazie i w przypadku istnienia danej osoby w bazie, aktualizuje jej dane, w przeciwnym wypadku dodaje nowy wpis do bazy danych.

Projekt dostępny jest na githubie:

https://github.com/Medokins/AiPO-project

Instrukcja Instalacji

Do poprawnego działania wymagany jest Python, zalecaną wersją jest 3.10.

Wymagane moduły można zainstalować za pomocą uruchomienia komendy w folderze projektu:

pip install -r requirements.txt

Dla systemów Windows wymagane jest również pobranie pliku wheel dla dlib: dlib-19.22.99-cp310-cp310-winpe_amd64.whl, oraz zainstalowanie go przy pomocy komendy:

pip install dlib-19.22.99-cp310-cp310-win_amd64.whl

Wymagane jest również pobranie plików z modelami:

- posePredictor.dat (https://github.com/davisking/dlib-models/blob/master/shape_predictor_68_face_landmarks_GTX.dat.bz2)
- 128Emb.dat (https://github.com/davisking/dlib-models/blob/master/dlib_face_recognition_resnet_model_v1.dat.bz2)

Opis Plików

main.py

Główny program z interfejsem użytkownika.

Funkcje:

- load_image()
    - Funkcja umożliwia użytkownikowi wybranie obrazu z systemu plików za pomocą interfejsu graficznego Tkinter. Otwiera okno dialogowe, które

pozwala użytkownikowi wybrać plik obrazu, i zwraca ścieżkę do wybranego pliku.

- read_image(image_path)
    - Wczytuje obraz z podanej ścieżki.
- update_database(image_path, name, surname)
    - Aktualizuje bazę danych osób na podstawie dostarczonego obrazu i informacji o osobie (imię, nazwisko). Najpierw sprawdza, czy plik bazy danych (database.csv) istnieje. Jeśli nie, tworzy nową bazę danych z odpowiednimi kolumnami. Następnie koduje twarz z dostarczonego obrazu. Jeśli twarz nie zostanie wykryta, funkcja informuje o tym i kończy działanie.
      Funkcja przeszukuje bazę danych w celu znalezienia już podobnej twarzy. Jeśli zostanie ona znaleziona, aktualizuje informacje o imieniu i nazwisku, jeśli są one dostępne. Jeśli twarz nie istnieje w bazie danych, funkcja zapisuje wyciętą twarz do katalogu detected_faces i dodaje nowy rekord do bazy danych z imieniem, nazwiskiem i ścieżką do pliku twarzy. Na koniec zapisuje zaktualizowaną bazę danych do pliku CSV.

Główna logika: Wczytuje obraz, przetwarza go, wyodrębnia tekst, a następnie aktualizuje bazę danych.

face_detection.py

Odpowiada za wykrywanie twarzy i porównywanie ich z istniejącymi w bazie danych.

Funkcje:

- get_face(image)
    - Funkcja wykrywa twarze na dostarczonym obrazie za pomocą detektora twarzy dlib. Jeśli zostanie wykryta co najmniej jedna twarz, funkcja zwraca lokalizację pierwszej znalezionej twarzy. W przeciwnym razie, zwraca None.
- encode_face(image):
    - Przekształca obraz twarzy w wektor cech, który może być użyty do porównywania twarzy. Najpierw wywołuje funkcję get_face w celu wykrycia lokalizacji twarzy na obrazie. Następnie, jeśli twarz zostanie wykryta, wykorzystuje model predykcji pozycji twarzy do identyfikacji punktów charakterystycznych na twarzy. Na podstawie tych punktów tworzy wycinek twarzy (face chip), który jest następnie przekształcany w wektor cech za pomocą modelu rozpoznawania twarzy dlib. Funkcja zwraca zarówno wektor cech, jak i lokalizację twarzy.
- get_similarity(encoding1, encoding2):

    - Funkcja oblicza odległość euklidesową między dwoma wektorami cech twarzy. Odległość ta jest miarą podobieństwa między dwiema twarzami – im mniejsza odległość, tym bardziej podobne są twarze.
- compare_encoded_faces(encoding1, encoding2):
    - Funkcja porównuje dwa wektory cech twarzy, aby określić, czy reprezentują one tę samą osobę. Oblicza odległość euklidesową między wektorami za pomocą funkcji get_similarity. Jeśli odległość jest mniejsza niż 0.6, funkcja zwraca True, wskazując, że twarze są wystarczająco podobne, aby mogły należeć do tej samej osoby. W przeciwnym razie zwraca False.
- save_cropped_face(image, face_location, save_dir): Zapisuje wykadrowaną twarz do określonego katalogu.
    - Zapisuje wycinek obrazu zawierający twarz do określonego katalogu. Najpierw wyodrębnia współrzędne twarzy, następnie tworzy wycinek obrazu zawierający tylko twarz. Jeśli katalog docelowy save_dir nie istnieje, funkcja tworzy go. Następnie zapisuje wycinek twarzy jako nowy plik JPEG w katalogu docelowym, używając unikalnej nazwy pliku. Na koniec funkcja zwraca ścieżkę do zapisanego pliku.

text_extraction.py

Zajmuje się przetwarzaniem obrazu i ekstrakcją tekstu, oraz identyfikacją imion i nazwisk.

Funkcje:

- preprocess_image(image_path):
    - Funkcja przetwarza obraz w celu przygotowania go do rozpoznawania tekstu. Proces ten obejmuje kilka kroków: konwersję obrazu na skalę szarości, zastosowanie rozmycia Gaussa, adaptacyjne progowanie w celu utworzenia obrazu binarnego, odwrócenie kolorów w obrazie binarnym oraz przekształcenie go z powrotem na obraz w kolorze BGR. Dzięki temu obraz staje się bardziej kontrastowy i wyraźny, co ułatwia dalsze rozpoznawanie tekstu przez algorytmy OCR.
- recognize_text(image): Rozpoznaje tekst na przetworzonym obrazie.
    - Wykorzystuje pipeline OCR do rozpoznawania i wyodrębniania tekstu z dostarczonego obrazu. Proces ten polega na analizie obrazu przez model OCR, który identyfikuje i zapisuje wszystkie rozpoznane słowa. Funkcja następnie zwraca listę odczytanych słów.
- find_closest_match(word, word_set)
    - Funkcja służy do znajdowania najbliższego dopasowania dla danego słowa w podanym zbiorze imion lub nazwisk. Wykorzystuje ona bibliotekę difflib, która porównuje słowo z elementami w zbiorze i

zwraca najbardziej podobne słowo, jeśli jego podobieństwo przekracza określony próg. Jest to przydatne w sytuacjach, gdy rozpoznane słowo może zawierać literówki lub drobne błędy, a chcemy znaleźć najbardziej prawdopodobne dopasowanie do imion lub nazwisk.

- extract_information(words)
    - Funkcja analizuje listę rozpoznanych słów w celu wyodrębnienia konkretnych informacji, takich jak imię, nazwisko i „specjalny numer”. Najpierw sprawdza, czy wśród słów znajdują się dokładne dopasowania do popularnych imion, nazwisk lub wzorców liczbowych. Jeśli nie znajdzie dokładnych dopasowań, próbuje znaleźć najbliższe dopasowania za pomocą funkcji find_closest_match. Funkcja zwraca wyodrębnione informacje, które mogą być następnie wykorzystane w innych częściach aplikacji, na przykład do aktualizacji bazy danych.

Za specjalny numer uznawany jest ciąg liczb o długości 4-10 znaków, może to być np. nr dokumentu.

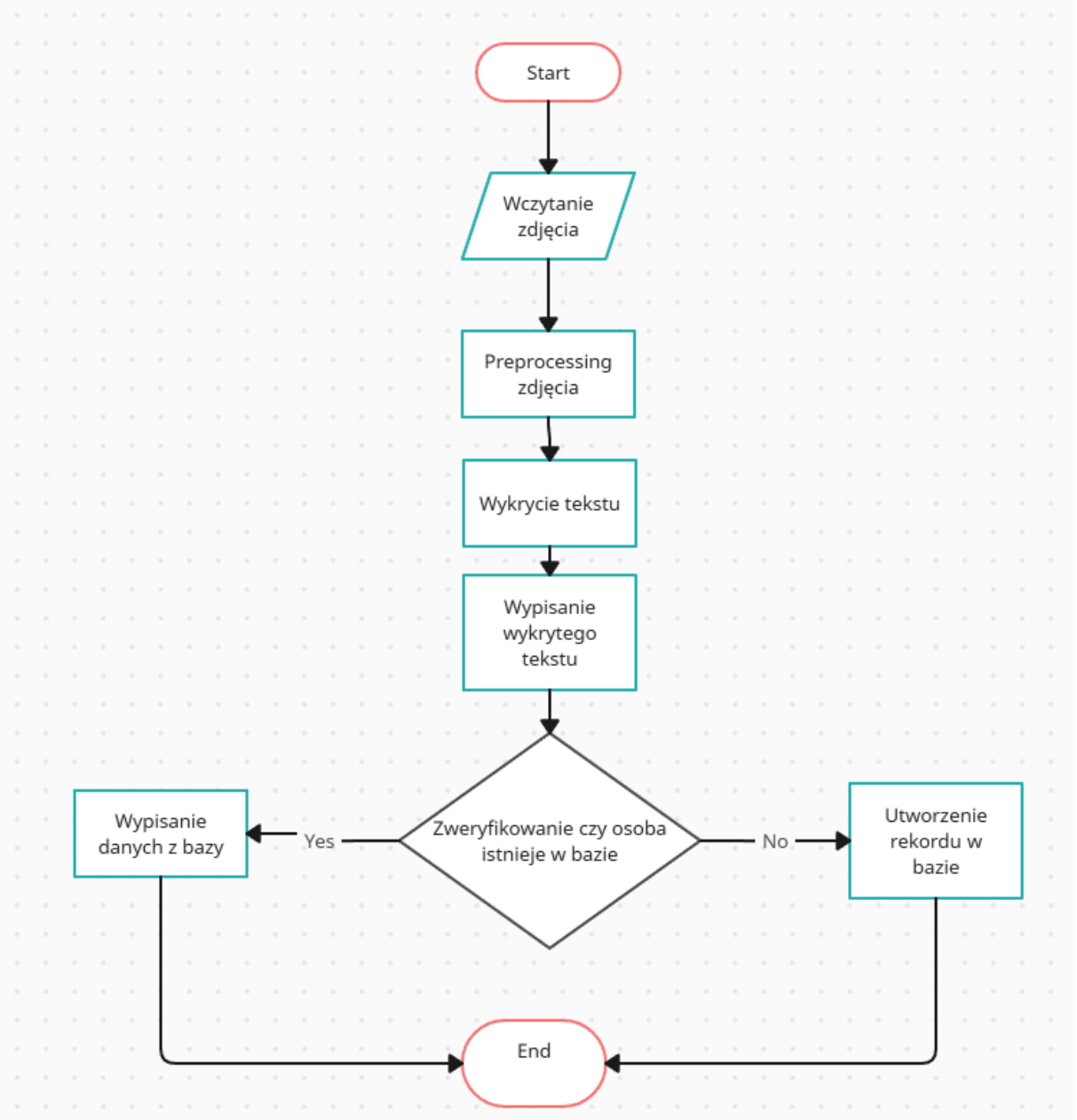
Uruchomienie programu

Program może być uruchomiony z katalogu projektu poprzez komendę:

python main.py

Po uruchomieniu pojawi się okno pozwalające na wybór pliku ze zdjęciem do przetworzenia.

Po wybraniu pliku zostanie on odpowiednio przetworzony, odczytane informacje są zapisywane do bazy oraz wyświetlane w terminalu. Program kończy działanie po wyświetleni rezultatu akcji.



Rysunek 1. Schemat działania programu

Dodatkowe Informacje

Baza Danych: database.csv przechowuje informacje o wykrytych twarzach i powiązanych z nimi danych.

Folder detected_faces: Przechowuje wykadrowane obrazy twarzy.

Pliki names_csv i polish_surnames.csv: wykorzystywane w celu wydobycia i ewentualnego poprawienia literówki w imieniu lub nazwisku.

Podział Pracy

1. Research technologii i wstępne założenia - Nikodem Szwast
2. Wykrywanie twarzy i zapisanie jej do bazy danych - Jakub Ochman
3. Enkodowanie twarzy - Jakub Wójcik
4. Wykrywanie tekstu - Krzysztof Tłuszcz
5. Korekcja tekstu (imion i nazwisk) - Marcel Loster
6. Identyfikacja specjalnego numeru - Mateusz Cichostępski
7. Konkatenacja fragmentów programu - Michał Pobuta
8. Logika wyszukiwania osoby w bazie danych - Szymon Nachlik