LABORATORIA SZTUCZNEJ INTELIGENCJI Sem. 5
2024/2025

Projekt zaliczeniowy laboratoria Sztucznej Inteligencji

Tytuł: Wykorzystanie silnika OCR Tesseract do wyciągania nieedytowalnego tekstu ze skanów dokumentów.

Adam Mazurek
numer albumu 77704

16.04.2025r.

1. Problem do rozwiązania

Celem projektu jest umożliwienie wyciągnięcia nieedytowalnego tekstu ze skanów dokumentów. Wyciągnięcie rozumiane jest jako umożliwienie zapisania treści dokumentu w formie edytowalnej bez konieczności ręcznego przepisywania jego treści do pliku. Umożliwi to zaawansowaną edycje tekstu w programie do edycji tekstu. Pomoże to użytkownikom szybko uzyskać dostęp do treści ze skanów, na przykład podręczników, które mogą wykorzystać w swojej pracy. Przykładem może być moja praca jako nauczyciel w technikum.

2. Rozwiązanie przy użyciu sztucznej inteligencji

Rozwiązanie będzie wykorzystywać silnik do rozpoznawania tekstu OCR Tesseract, który w praktyce wykorzystuje komponenty głębokiego uczenia LSTM. Cały proces będzie polegał na:

- Wczytaniu interesującego nas obrazu
- Preproccesing obrazu
- Przesłanie przeprocesowanego obrazu do OCR Tesseract
- Wypisanie wykrytego tekstu ze skanu

(OCR – Optical Character Recognition)
(LSTM – Long Short-Term Memory)

3. Wykorzystany model i architektura

W projekcie jest wykorzystany pretrenowany model OCR do rozpoznawania tekstu Tesseract.

- **Nazwa modelu:** Tesseract
- **Typ modelu:** Model oparty na architekturze LSTM, będącej odmianą rekurenycjnych sieci neuronowych RNN.
- **Źródło:** https://github.com/tesseract-ocr/tesseract/blob/main/README.md
- **Licencja:** Licencja Apache
- **Tytuł publikacji:** https://www.researchgate.net/publication/383772849_Study_of_Tesseract_OCR

4. Dane treningowe

OCR Tesseract od wersji 4.0 został wytrenowany z wykorzystaniem narzędzia tesstrain. Narzędzie to, generuje syntetyczne dane, na podstawie których Tesseract mógł się nauczyć odczytywać różne czcionki i języki. Ilość danych, które zostały wykorzystane do przetrenowania Tesseract, jest ogromna. Dla każdego języka mogło zostać wygenerowanych nawet kilka milionów syntetycznych próbek. Każda próbka to była para text plus grafika z tekstem. Dane były podzielone na zestaw treningowy i walidacyjny.

Link do tesstrain: https://github.com/tesseract-ocr/tesstrain

5. Wejścia/Wyjścia modelu

- **Wejścia modelu:** Wejściem do modelu jest obraz w odpowiednim formacie wrzucony przez użytkownika.
- **Kształt danych wejściowych:**
  - Tesseract nie ma ograniczeń co do wielkości wrzuconych obrazów, natomiast mają wpływ na wydajność pracy modelu. Za małe obrazy mogą być nieodczytane, a za duże, mogą być analizowane przez długi czas i wykorzystywać znaczne zasoby jednostki obliczeniowej.
  - Tesseract jest elastyczny jeżeli chodzi o kanały kolorów. W moim projekcie w preprocesingu zamieniam RGB na skale szarości, ale mógłbym wykorzystać na przykłąd RGB.
  - Obrazy mogą być w formatach PNG, JPG, BMP, TIFF, GIF, WebP i wiele innych popularnych formatów graficznych. Raz jeszcze, Tesseract jest bardzo elastycznym narzędziem i nie wybrzydza formatów.
- **Wyjścia modelu:** Wyjściem modelu jest tekst zeskanowany z obrazu. Można go skopiować i wkleić w całości do Worda i dokonać zaawansowanej edycji.
- **Kształt danych wyjściowych:** Tesseract, jak już pisałem wcześniej, jest bardzo elastyczny i możemy nadać odpowiedni kształt danych wyjściowych. Może być to typ danych typu string ale możemy też skonfigurować dane wyjściowe do bycia czystymi danymi binarnymi, albo ciągiem znaków z odpowiednimi dodatkami.

6. Wymagany preprocessing

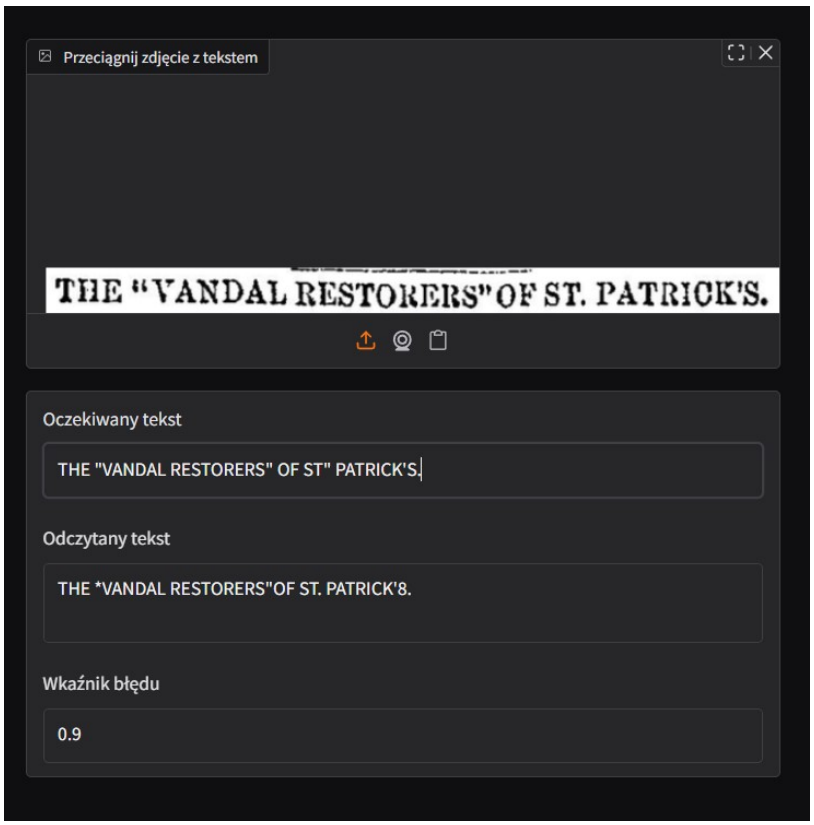
Aby zapewnić jak najlepszą jakość wykrywania tekstu, zaleca się wykonanie preprocessingu obrazów. W moim projekcie wykonałem następujący preprocessing:

- Przez to, że używam cv2, dokonuje konwersji RGB na BGR
- Konwersja koloru do skali szarości

W przypadku mojego projektu taki preprocessing jest wystarczający, ale gdybym potrzebował jeszcze bardziej polepszyć jakość obrazu dla silnika Tesseract, to mógłbym usunąć szumy.

7. Jak będzie określana dokładność modelu?

Dokładność modelu będzie określona na podstawie wskaźnika dokładności rozpoznanego tekstu. Wyliczenie wskaźnika polegać będzie na porównaniu tekstu rozpoznanego z oczekiwanym. Przykład będzie załączony w projekcie. W prezentacyjnym demo została dodana zakładka, która umożliwia samodzielne sprawdzenie dokładności modelu.



Zdj. 1 – Przykład badania dokładności modelu

8. Język programowania i wykorzystywane biblioteki

Implementacja kodu będzie wykonana w języku Python. Demo zostanie wykorzystane w Gradio. Biblioteki wykorzystane w projekcie:

1. gradio – Biblioteka Python do szybkiego tworzenia UI
2. pytesseract – Wrapper dla silnika OCR Tesseract w Python
3. cv2 – OpenCV (Open Source ComputerVision Library) do zaawanowanej obsługi obrazów
4. numpy – Podstawowa biblioteka do obliczeń naukowych
5. difflib – Biblioteka do porównywania tekstów

9. Rodzaj inferencji

W moim projekcie inferencja jest wykonywana lokalnie na jednostce, która będzie kompilować kod.

10. Możliwość dalszego rozwoju projektu

Projekt po dodatkowych poprawkach/ulepszeniach może okazać się przydatnym narzędziem w mojej pracy jako nauczyciel w technikum. W swojej pracy korzystam ze skanów podręczników do nauki elektryki i elektroniki i edytowanie tekstu z takiego podręcznika może znacząco wspomóc moją prace dydaktyczną. Edycje tekstu mógłbym na przykład wykorzystać przy tworzeniu materiałów do druku dla uczniów. Kolejną moją uwagą jest możliwość wyuczenia Tesseract w taki sposób, aby mógł odczytywać tekst prac domowych uczniów, które później skrypt sprawdzałby automatycznie. Moja praca ograniczyłaby się tylko do wrzucenia zdjęć.