Politechnika Warszawska

Wydział Mechaniczny Energetyki i Lotnictwa

OBLICZENIA INŻYNIERSKIE W CHMURZE

System masowego przetwarzania aktów prawnych (.docx) do strukturalnego formatu (.csv)

inż. Edgar Jończyk

320866

WARSZAWA 2026

SPIS TREŚCI

1. Wstęp

Celem projektu było opracowanie i implementacja zautomatyzowanego narzędzia, służącego do masowego przetwarzania nieustrukturyzowanych tekstowych dokumentów prawa polskiego (w formacie .docx) do postaci ustrukturyzowanej (format .csv). Projekt powstał w odpowiedzi na potrzebę optymalizacji czasochłonnego procesu ręcznej analizy aktów prawnych i specyfikacji technicznych.

Głównym założeniem inżynierskim było stworzenie aplikacji przenośnej i niezależnej od systemu operacyjnego hosta. W tym celu wykorzystano całe środowisko wykonawcze w kontenerze Docker. Umożliwiło to symulację gotową do wdrożenia w chmurze obliczeniowej (Azure Container Instances), zapewniając powtarzalność wyników oraz izolację procesu przetwarzania wsadowego (batch processing).

2. Opis zagadnienia i charakterystyka technologiczna

Podstawowym wyzwaniem technicznym w projekcie była konwersja danych nieustrukturyzowanych (prawne dokumenty tekstowe w formacie .docx) do formatu ustrukturyzowanego, kompatybilnego z systemami bazodanowymi. Akty prawne charakteryzują się specyficzną hierarchią (artykuły, ustępy, paragrafy), której rozpoznanie wymagało zastosowania zaawansowanych metod parsowania tekstu.

W ramach projektu zaimplementowano algorytm oparty na wyrażeniach regularnych, pozwalający na precyzyjną identyfikację jednostek redakcyjnych tekstu niezależnie od ich formatowania wizualnego. Dodatkowo, system został wyposażony w moduł integracji z zewnętrznymi API (Google Translate) w celu automatycznego tłumaczenia wyekstrahowanych treści, co symuluje rzeczywiste scenariusze pracy w środowiskach międzynarodowych.

Do realizacji zadania wybrano:

Język Python 3.12:

Wybrany jako wiodąca technologia w dziedzinie Data Science i przetwarzania tekstu. Zastosowano go w wersji slim w celu optymalizacji rozmiaru obrazu wynikowego.

Docker & Konteneryzacja:

Środowisko uruchomieniowe zostało zdefiniowane w pliku Dockerfile. Zastosowanie konteneryzacji pozwoliło na wyeliminowanie problemów z zależnościami systemowymi oraz zapewniło pełną przenośność rozwiązania między chmurą a środowiskiem lokalnym.

Kluczowe biblioteki:

python-docx: Do niskopoziomowej analizy struktury plików Word.

langdetect i deep-translator: Do detekcji języka i tłumaczenia maszynowego.

3. Struktura folderów i plików projektu

Projekt został zorganizowany w przejrzystą strukturę katalogów, co ułatwia zarządzanie kodem oraz danymi. Każdy plik pełni ściśle określoną rolę w procesie przetwarzania. Poniżej przedstawiono opis poszczególnych elementów projektu:

input_files - Folder wejściowy. Jest to miejsce, w którym użytkownik umieszcza dokumenty .docx (akty lub rozporządzenia polskiego prawa) przeznaczone do przetworzenia. Program automatycznie skanuje ten katalog w poszukiwaniu nowych plików.

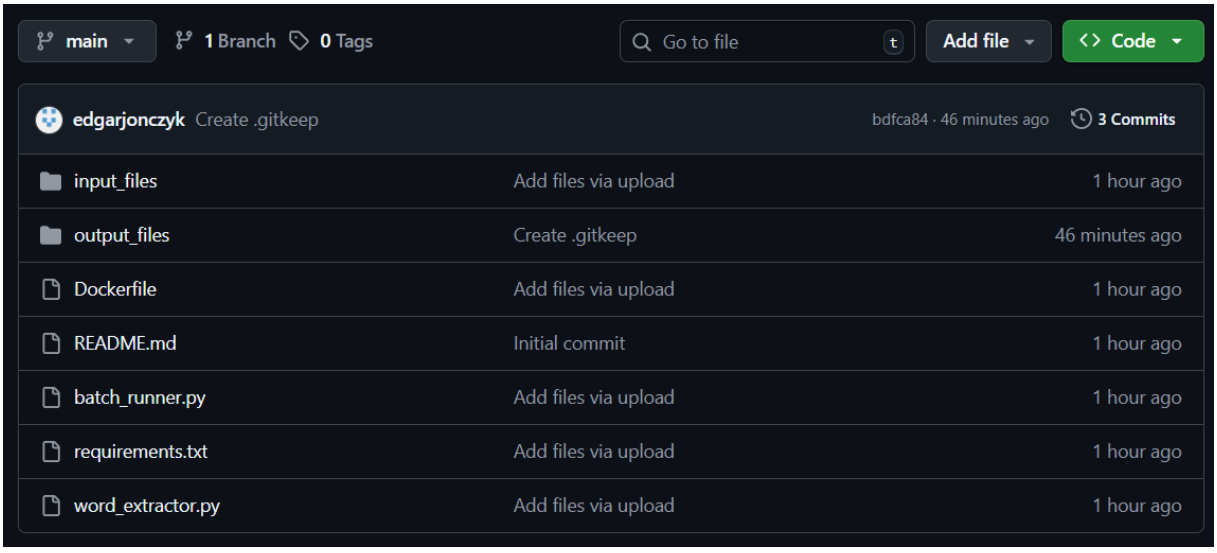
output_files - Folder wynikowy. Tutaj trafiają wygenerowane pliki .csv. Dzięki zastosowaniu mechanizmu wolumenów w Dockerze, pliki te są od razu widoczne na komputerze użytkownika, bez konieczności ręcznego kopiowania ich z wnętrza kontenera.

Dockerfile - Plik konfiguracyjny środowiska. Zawiera „przepis" dla Dockera: jaki system operacyjny pobrać (Python Slim), jakie narzędzia systemowe doinstalować i jak uruchomić aplikację. Gwarantuje, że program zadziała identycznie na każdym komputerze.

batch_runner.py - Główny skrypt uruchomieniowy. Odpowiada za sprawdzenie folderów, znalezienie plików do przetworzenia i uruchomienie procesu dla każdego z nich po kolei.

requirements.txt - Lista niezbędnych bibliotek Python (zależności). Dzięki temu plikowi Docker wie, że musi zainstalować pakiety takie jak python-docx czy deep-translator przed uruchomieniem programu.

word_extractor.py - Biblioteka zawierająca „mózg" operacji. To tutaj znajdują się funkcje odpowiedzialne za czytanie pliku Word, wykrywanie paragrafów, czyszczenie tekstu oraz łączenie się z translatorem.



Rysunek 1 - Struktura folderów i plików w projekcie

4. Architektura systemu i opis implementacji

Program został zaprojektowany w sposób modułowy - czyli podzielono go na dwie oddzielne części: jedną, która zarządza plikami, i drugą, która zajmuje się wyciąganiem tekstu. Dzięki temu kod jest czytelny i łatwiejszy w naprawie. Całość działa wewnątrz kontenera Docker, co gwarantuje, że program uruchomi się tak samo na każdym komputerze.

a) Skrypt zarządzający (batch_runner.py)

Jest to główny plik, który uruchamia całą machinę. Jego zadanie jest proste: sprawdza folder wejściowy (input_files) i szuka w nim wszystkich plików Word (.docx). Jeśli folder z wynikami nie istnieje, skrypt sam go utworzy. Co ważne, skrypt posiada zabezpieczenie przed błędami. Jeśli jeden z plików okaże się uszkodzony, program wypisze błąd w konsoli, ale nie przerwie pracy – po prostu przejdzie do kolejnego dokumentu.

b) Moduł wyciągania danych (word_extractor.py)

To tutaj dzieje się główna praca. Program działa w kilku krokach:

- Szukanie treści:

Skrypt czyta dokument akapit po akapicie. Używa specjalnych wzorców (tak zwanych wyrażeń regularnych), aby znaleźć fragmenty zaczynające się od słów takich jak „Art.” lub symbolu „§”. Dzięki temu wyciąga tylko to, co jest aktem prawnym, pomijając zbędne ozdobniki.

- Czyszczenie:

Zanim tekst trafi do pliku wynikowego, jest czyszczony. Program automatycznie usuwa na przykład teksty w nawiasach kwadratowych, które często są tylko technicznymi dopiskami.

- Inteligentne tłumaczenie:

Tłumaczenie długich tekstów bywa problematyczne, bo serwisy takie jak Google Translate mają limity znaków. Dlatego zastosowano tu autorski mechanizm, który dzieli długi tekst na mniejsze kawałki (do 4000 znaków). Co istotne - program nie ucina tekstu w połowie zdania, ale szuka kropki lub końca akapitu, żeby tłumaczenie miało sens.

c) Obsługa danych (Wejście/Wyjście)

Aplikacja działa na zasadzie „przetwórz i zapomnij”. Nie przechowuje danych wewnątrz siebie na stałe. Pobiera pliki z folderu, który podłączamy na początku (input_files), przetwarza je i zapisuje gotowe tabele (CSV) w folderze wyjściowym (output_files). Dzięki temu po wyłączeniu programu nie zostają żadne zbędne śmieci, a wyniki mamy od razu na swoim dysku.

5. Prezentacja działania i otrzymane wyniki

W tym rozdziale przedstawiono praktyczny test działania systemu. Cały proces - od przygotowania „wirtualnego pudełka” (obrazu Docker) aż po uzyskanie gotowych danych.

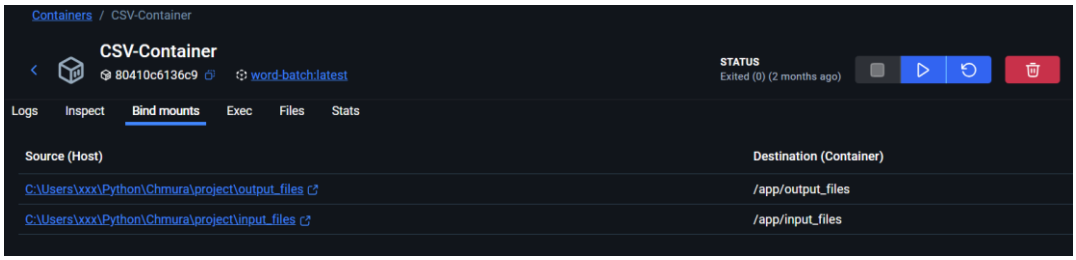
Krok 1: Budowanie środowiska (Docker Build)

Pierwszym etapem było zbudowanie obrazu na podstawie pliku Dockerfile. Jest to moment, w którym Docker czyta nasz „przepis”, pobiera system i instaluje wszystkie wymagane biblioteki (takie jak python-docx czy deep-translator).

Krok 2: Uruchomienie przetwarzania wsadowego

Następnie uruchomiono kontener, podłączając do niego folder z przykładowymi dokumentami prawnymi (Rysunek 2 i Rysunek 3).

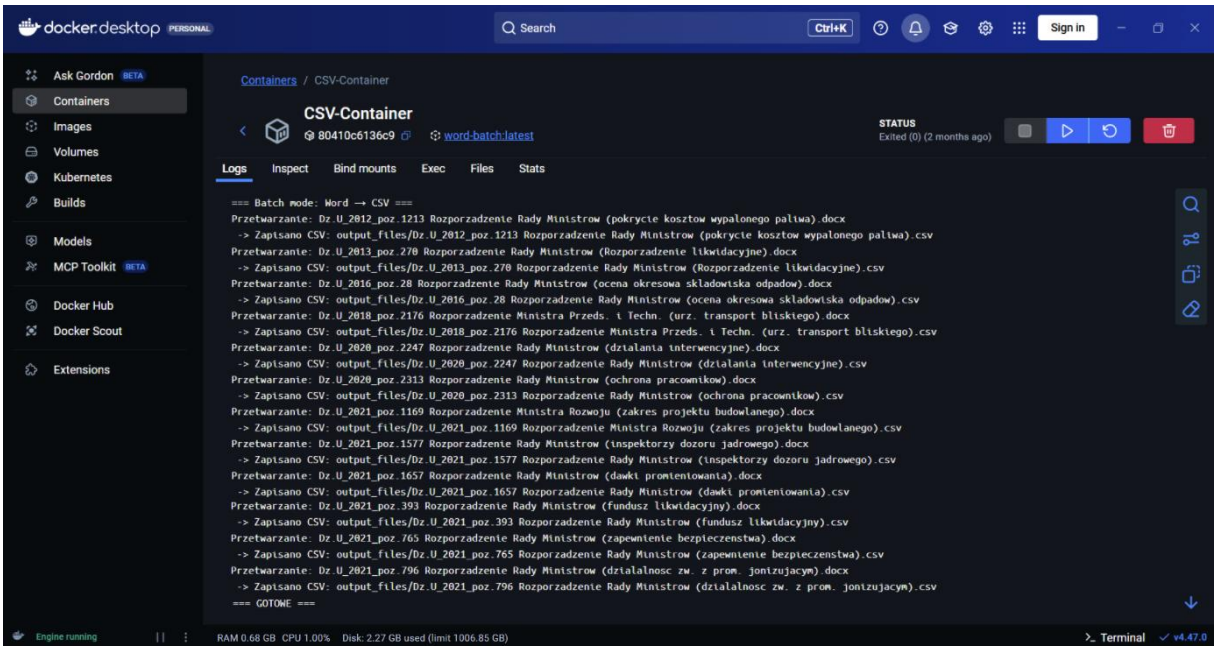
Rysunek 2 - Lista przykładowych aktów prawnych użytych w projekcie



Rysunek 3 - Zaimplementowane foldery wejściowy i wyjściowy w Dockerze

Krok 3: Działanie

Aplikacja natychmiast rozpoczęła skanowanie. W konsoli (Rysunek 4) widzimy logi, które program Dockera wypisuje na żywo: informuje nas, jaki plik aktualnie przetwarza oraz potwierdza zapisanie wyniku. Dzięki temu użytkownik wie, że system nie zawiesił się, lecz pracuje nad kolejnymi plikami.



Rysunek 4 - Konsola Dockera z przeprocesowanymi plikami wsadowymi

Krok 4: Analiza wyników (Plik CSV)

Finalnym efektem pracy programu jest plik .csv, który pojawił się w folderze wynikowym. Na rysunku 4 zaprezentowano jego zawartość. Widać wyraźnie, że nieuporządkowany tekst z pliku Word został zamieniony na czytelną tabelę. System poprawnie rozpoznał strukturę dokumentu:

Column1	Column2	Column3	Column4	Column5	Column6	Column7
Description	Position	Summary	Primary Languag	Description Translated	Position II	Position + Summary
1. Ustawa określa:1) działalność w zakresie pokojowego wykorzystywania energii atomowej :	Art. 1	Dz.U_2024_poz.1277 Ustawa Prawo Atomowe	Polish	1. The Act specifies:1) activities in the peaceful use of atomic energy related to actual and pot	Art. 1	Art. 1 Dz.U_2024_poz.1277 Ustawa Prawo Atomowe
Wykonywanie działalności, o której mowa w art. 1 ust. 1 pkt 1 i ust. 3, jest dopuszczalne po z	Art. 2	Dz.U_2024_poz.1277 Ustawa Prawo Atomowe	Polish	Carrying out the activities referred to in Art. 1 section 1 point 1 and section 3, is permissible	Art. 2	Art. 2 Dz.U_2024_poz.1277 Ustawa Prawo Atomowe
W rozumieniu niniejszej ustawy użyte określenia oznaczają:1) audyt kliniczny - systematyczn	Art. 3	Dz.U_2024_poz.1277 Ustawa Prawo Atomowe	Polish	For the purposes of this Act, the terms used mean:1) clinical audit - a systematic control or re	Art. 3	Art. 3 Dz.U_2024_poz.1277 Ustawa Prawo Atomowe
1. Wykonywanie działności związanej z narażeniem polegającej na:1) wytwarzaniu, przetw	Art. 4	Dz.U_2024_poz.1277 Ustawa Prawo Atomowe	Polish	1. Performing activities involving exposure consisting of:1) production, processing, storage, t	Art. 4	Art. 4 Dz.U_2024_poz.1277 Ustawa Prawo Atomowe
1. Wniosek o wydanie zezwolenia na wykonywanie działalności, o której mowa w art. 4 ust. 1	Art. 5	Dz.U_2024_poz.1277 Ustawa Prawo Atomowe	Polish	1. Application for a permit to perform the activities referred to in Art. 4 section 1, includes:1)	Art. 5	Art. 5 Dz.U_2024_poz.1277 Ustawa Prawo Atomowe

Rysunek 5 - Przykładowy wynik pliku CSV dla przeprocesowanego pliku Word dla Prawa Atomowego

Omówienie pól wynikowego pliku CSV:

Description: Zawiera treść artykułu.

Position i Position II: Wskazuje numer prawny (np. Art. 1, § 1).

Summary: Tytuł aktu prawnego, ustawy lub rozporządzenia

Translated: Zawiera automatyczne tłumaczenie treści na język angielski.

Description Translated: Zawiera przetłumaczoną treść artykułu (analogiczne tłumaczenie do pola Description).

Position+Summary: Połączone pola Position oraz Summary dla uspójnienia całej nazwy danego artykułu i nadania mu unikalności.

6. Podsumowanie i wnioski

Projekt zakończył się sukcesem - udało się stworzyć w pełni funkcjonalne narzędzie, które automatyzuje żmudny proces grupowego przepisywania danych z dokumentów Word.

Kluczowe wnioski z realizacji projektu:

Siła konteneryzacji:

Dzięki Dockerowi nie trzeba było tracić czasu na konfigurację środowiska czy instalację skomplikowanych bibliotek systemowych na komputerze hosta. Program jest w 100% przenośny - ten sam kod zadziała identycznie na laptopie studenta, jak i na serwerze w chmurze Azure.

Oszczędność czasu:

Przetwarzanie wsadowe pozwala na konwersję setek dokumentów w czasie, w którym człowiek przetworzyłby zaledwie kilka stron. Jest to ogromne usprawnienie w pracy z dokumentacją techniczną lub prawną.

Gotowość na chmurę:

Aplikacja została napisana i jest zamknięta w kontenerze - nie przechowuje danych na stałe (korzysta z zewnętrznych folderów), jest gotowa do wdrożenia w usługach chmurowych, takich jak Azure Container Instances, bez konieczności wprowadzania zmian w kodzie.

Podsumowując, projekt pokazuje, jak proste narzędzia inżynierskie (Python + Docker) mogą rozwiązywać realne problemy biznesowe, eliminując powtarzalną pracę ręczną.

Repozytorium projektu:

Kompletny kod źródłowy oraz wszystkie potrzebne pliki wraz z raportem znajdują się pod adresem:

https://github.com/edgarjonczyk/Obliczenia_inzynierskie_w_chmurze_EdgarJ