Studia i Materiały Informatyki Stosowanej, Tom 18, Nr 1, 2026
str. 11-15

SYSTEM WSPOMAGAJĄCY GROMADZENIE I PRZETWARZANIE DANYCH Z WYKORZYSTANIEM MODELI JĘZYKOWYCH I TECHNOLOGII OCR

Łukasz Łapiak, Piotr Kotlarz

Kazimierz Wielki University, Faculty of Computer Science
Kopernika 1, 85-074 Bydgoszcz
e-mail: lukasz.lapiak@student.ukw.edu.pl

Abstract: *This article presents a comprehensive system supporting real-time collection and processing of visual data using OCR technology and language models. The research compares six text recognition tools: local engines (Tesseract, EasyOCR), external services (OCR.space), and multimodal language models (GPT-4, Gemini 1.5 Flash, Claude 3 Haiku). The study demonstrates that the effectiveness of the technology depends on the data type. Multimodal language models achieved significantly higher accuracy in analyzing complex data (such as handwriting), whereas for standard digital text, local OCR solutions offered comparable precision with significantly faster processing times. The Flask-based web application with MySQL enables efficient data management. Levenshtein distance metric was used for accuracy measurement. Results indicate the validity of a hybrid approach, integrating the speed of traditional OCR with the semantic capabilities of modern AI models.*

Keywords: *OCR, Language models, Text analysis, Image processing, Flask, MySQL, Tesseract, GPT-4, Gemini, Levenshtein*

SYSTEM SUPPORTING DATA COLLECTION AND PROCESSING USING LANGUAGE MODELS AND OCR TECHNOLOGY

Streszczenie: *Niniejszy artykuł przedstawia kompleksowy system wspomagający gromadzenie i przetwarzanie danych wizualnych w czasie rzeczywistym z wykorzystaniem technologii OCR oraz modeli językowych. Badania porównują sześć narzędzi rozpoznawania tekstu: lokalne silniki (Tesseract, EasyOCR), zewnętrzne usługi (OCR.space) oraz multimodalne modele językowe (GPT-4, Gemini 1.5 Flash, Claude 3 Haiku). Analiza wykazała, że skuteczność technologii jest ściśle zależna od rodzaju materiału źródłowego. Modele językowe osiągnęły znaczącą przewagę w rozpoznawaniu pisma odręcznego i treści złożonych, podczas gdy dla standardowego tekstu cyfrowego tradycyjne silniki OCR zaoferowały porównywalną precyzję przy znacznie wyższej szybkości przetwarzania. Aplikacja webowa oparta na Flask i MySQL umożliwia efektywne zarządzanie danymi. Do pomiaru dokładności wykorzystano metrykę odległości Levenshteina. Wyniki wskazują na zasadność stosowania podejścia hybrydowego, integrującego wydajność lokalnego OCR z możliwościami semantycznymi modeli AI.*

Słowa kluczowe: *OCR, Modele językowe, Analiza tekstu, Przetwarzanie obrazu, Flask, MySQL, Tesseract, GPT-4, Gemini, Odległość Levenshtein*

1. Wprowadzenie

Rozwój technologii sztucznej inteligencji i uczenia maszynowego znacząco wpłynął na możliwości automatyzacji procesów przetwarzania danych wizualnych. Optyczne rozpoznawanie znaków (OCR) stanowi kluczową technologię w digitalizacji dokumentów, jednak tradycyjne podejścia napotykają istotne ograniczenia w interpretacji niestandardowych czcionek oraz złożonych układów graficznych [6]. Rosnące zapotrzebowanie na uniwersalne narzędzia ekstrakcji tekstu sprawia, że poszukiwanie nowych metod analizy stało się priorytetem w inżynierii danych.

Celem niniejszego artykułu jest przedstawienie i ewaluacja kompleksowego systemu wspomagającego gromadzenie danych, wykorzystującego zarówno tradycyjne algorytmy OCR, jak i nowoczesne modele językowe. Integracja odmiennych podejść technologicznych pozwala na rzetelną

ocenę ich wydajności w kontekście praktycznych zastosowań biznesowych.
W pracy przyjęto hipotezę badawczą zakładającą, że skuteczność ekstrakcji informacji jest ściśle zależna od specyfiki materiału źródłowego. Przewiduje się, że multimodalne modele językowe (LLM) wykażą znaczącą przewagę w rozpoznawaniu pisma odręcznego i treści o złożonej strukturze, podczas gdy tradycyjne silniki OCR zachowają wyższą efektywność obliczeniową i porównywalną precyzję w analizie standardowego tekstu cyfrowego.
Zaprojektowany system składa się z aplikacji webowej opartej na frameworku Flask, bazy danych MySQL oraz interfejsu umożliwiającego dynamiczny wybór metody przetwarzania. Przeprowadzone badania porównawcze obejmują sześć narzędzi analizy obrazu, co pozwala na weryfikację ich użyteczności w zróżnicowanych scenariuszach [6, 7, 8]. Do obiektywnej oceny jakości rozpoznawania tekstu zastosowano metrykę odległości Levenshteina, pozwalającą na kwantyfikację błędów względem wzorca.

Tabela 1 przedstawia porównanie trzech kluczowych modeli językowych wykorzystanych w badaniu pod względem daty treningu, limitów API oraz kosztów przetwarzania.

Model	Data treningu	RPM	Koszt*	Dostępność
GPT-4 Turbo	Koniec 2022	500	$10/$30	Wycofany
Gemini 1.5 Flash	Maj 2023	15	$0.075/$0.30	Wycofany
Claude 3 Haiku	Sierpień 2023	50	$0.25/$1.25	Tak (Legacy)

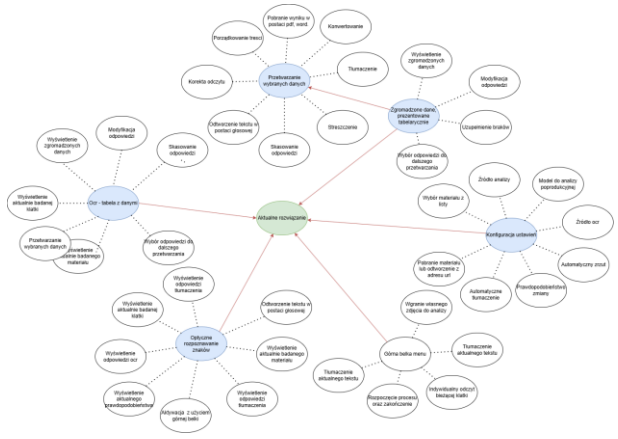
** Cena za milion tokenów (wejście/wyjście) w USD, 2024 rok.*

Tabela 1. Porównanie wybranych modeli językowych (opracowanie własne na podstawie [3, 4, 5]).

2. Materiał i metody

Materiał badawczy podzielono na dwa zbiory danych. Główną próbkę (Zbiór A) stanowiło nagranie wideo w rozdzielczości 1920×1080 pikseli, zawierające tekst drukowany (Lyric Video) [9]. W procesie ekstrakcji danych zastosowano algorytm próbkowania z interwałem czasowym 6 sekund, wyposażony w mechanizm detekcji zmian. System wstępnie przetwarzał klatkę i porównywał jej zawartość z poprzednim odczytem – próbka była zapisywana do bazy badawczej wyłącznie wtedy, gdy wykryta różnica przekraczała zdefiniowany próg 40%. Pozwoliło to na eliminację klatek statycznych i duplikatów, redukując zbiór do 39 kluczowych, unikalnych próbek reprezentatywnych dla całego utworu.
Drugi zbiór (Zbiór B) obejmował obrazy zawierające pismo odręczne oraz tekst o średniej jakości, przygotowany w celu weryfikacji skuteczności narzędzi w warunkach niestandardowych.
Do końcowej oceny jakości rozpoznawania wykorzystano metrykę odległości Levenshteina [10]. Została ona zastosowana do porównania wyników wygenerowanych przez poszczególne narzędzia (zapisanych próbek) względem tekstu referencyjnego (wzorca). Metryka ta określa minimalną liczbę operacji edycji potrzebnych do przekształcenia jednego ciągu znaków w drugi – im niższa wartość końcowa, tym wyższa precyzja odwzorowania tekstu przez dane narzędzie. Badanie przeprowadzono za pomocą autorskiego skryptu Python [11], a wizualizację danych wykonano biblioteką Matplotlib [12].

Rysunek 1 przedstawia architekturę systemu w formie diagramu przypadków użycia, ilustrującego główne funkcjonalności aplikacji oraz interakcje użytkownika z systemem.



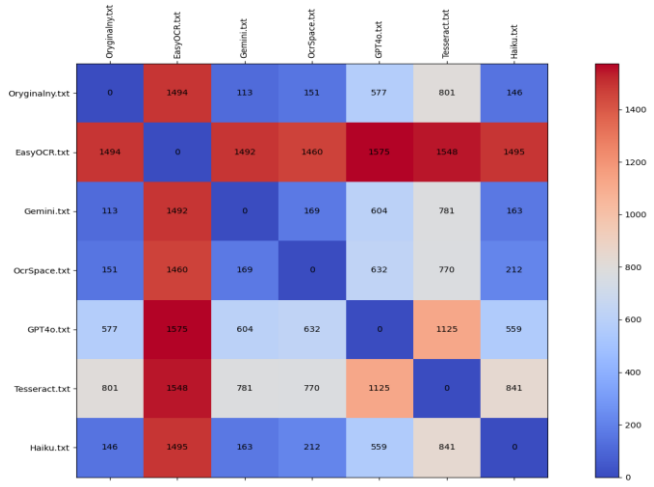
Rysunek 1. Architektura systemu – diagram przypadków użycia.

3. Wyniki

Wyniki badań potwierdziły hipotezę badawczą, wykazując znaczącą przewagę multimodalnych modeli językowych nad tradycyjnymi silnikami OCR. Dla materiału zawierającego tekst drukowany, najlepszą dokładność

osiągnęły kolejno: OCR.space z odległością Levenshteina 39, Tesseract (42), Gemini (52), Claude 3 Haiku (147) oraz EasyOCR (269). Model GPT-4 uzyskał najgorszy wynik (544) z powodu problemów z odmową przetwarzania niektórych klatek, które algorytm uznał za potencjalnie naruszające prawa autorskie.
W przypadku tekstu pisanego odręcznie różnice były jeszcze bardziej wyraźne. Modele językowe Gemini i Claude 3 Haiku osiągnęły najlepsze wyniki, rozpoznając tekst z niewielkimi odstępstwami od oryginału. GPT-4, mimo sporadycznych drobnych błędów, dostarczył czytelny i zrozumiały tekst. Lokalne silniki OCR całkowicie zawiodły – Tesseract uzyskał odległość 801, a EasyOCR 1494, co oznacza niemal całkowity brak zdolności rozpoznania pisma odręcznego.

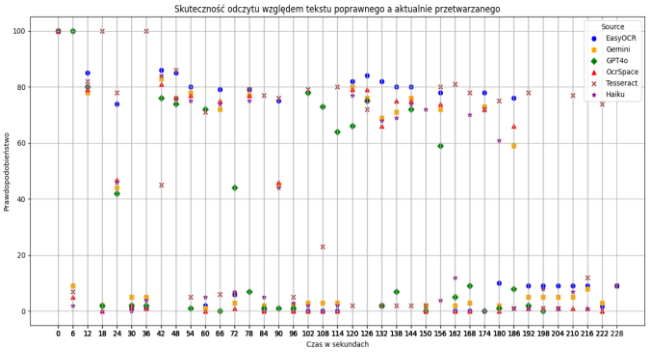
Rysunek 2 przedstawia mapę ciepła wizualizującą dokładność rozpoznawania tekstu dla wszystkich badanych narzędzi. Im ciemniejszy (bardziej niebieski) kolor, tym wyższa dokładność, co odpowiada niższym wartościom odległości Levenshteina.



Rysunek 2. Mapa ciepła porównania dokładności narzędzi OCR.

Analiza dynamiczna wykazała różnice w stabilności rozpoznawania. Lokalne silniki charakteryzowały się szybszym przetwarzaniem, jednak modele językowe wykazały wyższą stabilność wyników. Model GPT-4 napotkał problemy z odmową przetwarzania niektórych klatek z powodu polityki praw autorskich, co skutkowało niekompletnymi wynikami analizy.

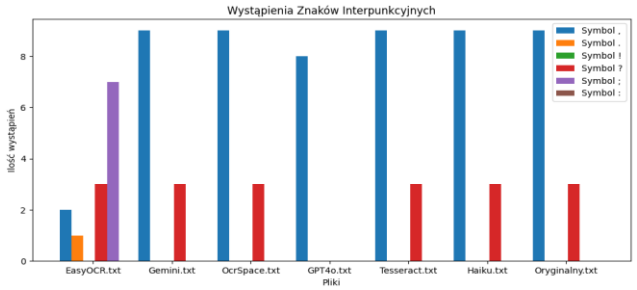
Rysunek 3 ilustruje dynamiczną analizę stabilności rozpoznawania w czasie trwania materiału wideo. Oś odciętych (X) reprezentuje czas w sekundach, natomiast oś rzędnych (Y) obrazuje skuteczność odczytu (znormalizowaną do skali 0-100 na podstawie odwrotności dystansu Levenshteina).
Należy zwrócić uwagę na nieciągłość punktów pomiarowych, która jest bezpośrednim skutkiem zastosowanej metodyki próbkowania. Algorytm analizował strumień wideo w interwałach 6-sekundowych, jednak decyzja o przekazaniu klatki do silników OCR była podejmowana warunkowo. Zastosowano filtr detekcji zmian z progiem czułości 40% – oznacza to, że próbka była przetwarzana i nanoszona na wykres tylko wtedy, gdy jej zawartość wizualna zmieniła się znacząco względem poprzedniego odczytu. Takie podejście pozwoliło wyeliminować z analizy klatki statyczne oraz momenty przejść (wygaszenia ekranu), koncentrując się na momentach, w których na ekranie pojawiała się nowa partia tekstu.



Rysunek 3. Dynamiczna analiza czasowa stabilności rozpoznawania dla wszystkich narzędzi dla próbki pisma.

Analiza zawartości tekstowej wykazała, że modele językowe lepiej zachowują strukturę tekstu i znaki interpunkcyjne. Gemini, OCR.space, Tesseract i Haiku poprawnie rozpoznały wszystkie 9 przecinków i 3 znaki zapytania w tekście referencyjnym, podczas gdy GPT-4 dodał dodatkową kropkę, a EasyOCR znacząco odbiegał od oryginału.

Rysunek 4 przedstawia analizę wystąpień znaków interpunkcyjnych w rozpoznanym tekście. Wykres słupkowy porównuje liczbę poprawnie zidentyfikowanych przecinków, kropek i znaków zapytania dla każdego narzędzia.



Rysunek 4. Analiza wystąpień znaków interpunkcyjnych.

4. Dyskusja

Wyższa dokładność modeli językowych wynika z wykorzystania głębokiego uczenia i mechanizmów uwagi, które pozwalają na zrozumienie kontekstu tekstu, a nie tylko rozpoznawanie pojedynczych znaków. Architektura transformerów, na której oparte są modele GPT-4-Turbo, Gemini i Claude Haiku, umożliwia analizę długoterminowych zależności w tekście, co jest szczególnie istotne przy rozpoznawaniu tekstu pisanego odręcznie lub w niestandardowych formatach [14, 15].

Koszty operacyjne stanowią istotny czynnik w wyborze metody. Podczas gdy lokalne silniki nie generują kosztów związanych z tokenizacją, modele językowe wymagają płatności za wykorzystanie API. Według danych producenta GPT-4 Turbo kosztuje $10 za milion tokenów wejściowych i $30 za wyjściowe, Gemini 1.5 Flash oferuje bardziej konkurencyjne ceny ($0.075/$0.30 za milion tokenów), Claude 3 Haiku oferuje ($0.25/$1.25 za milion przetworzonych tokenów) [3, 4, 5]. Dla zastosowań wymagających przetwarzania dużych wolumenów danych, konieczne jest uwzględnienie tego aspektu w analizie ekonomicznej. Ograniczenia badania obejmują wykorzystanie materiału cyfrowego o wysokiej jakości (1920×1080px). Przyszłe badania powinny objąć analizę dokumentów historycznych o różnym stopniu degradacji, materiałów wielokolumnowych oraz tekstów w językach wykorzystujących nietypowe style pisma. Dodatkowo, warto zbadać hybrydowe podejście łączące szybkość lokalnych silników optycznego rozpoznawania znaków z dokładnością modeli językowych, wykorzystując OCR jako pierwszy etap przetwarzania, a modele AI do weryfikacji i korekty wyników. Praktyczne zastosowania systemu obejmują automatyzację digitalizacji dokumentów, materiałów prezentowanych np. na transmisji na żywo, tłumaczenie informacji w czasie rzeczywistym oraz analizę treści multimedialnych. Zaimplementowana aplikacja webowa (Flask + MySQL) umożliwia skalowalne przetwarzanie danych z możliwością integracji z istniejącymi systemami przedsiębiorstw [1, 2, 13].

5. Implementacja interfejsu użytkownika

Zaprojektowany system został wyposażony w graficzny interfejs użytkownika (GUI), którego architektura opiera się na wyraźnym podziale funkcjonalności. Zastosowano układ z ciemnym, lewostronnym paskiem nawigacyjnym (sidebar), który kontrastuje z jasnym obszarem roboczym. Taki zabieg wizualny pozwala na szybką identyfikację dostępnych modułów i płynne przechodzenie między etapami badania: od wstępnej konfiguracji, przez procesy optycznego rozpoznawania znaków, aż po analizę danych.

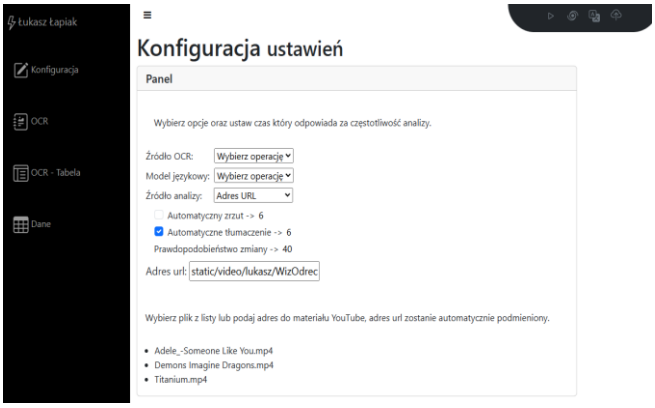
Centralnym punktem sterowania procesem badawczym jest panel konfiguracyjny (Rysunek 5). To tutaj użytkownik definiuje kluczowe parametry eksperymentu przed rozpoczęciem przetwarzania. Interfejs umożliwia dynamiczny wybór silnika OCR, modelu językowego oraz źródła sygnału wideo (plik lokalny lub adres URL).

Szczególne znaczenie dla weryfikacji przyjętej metodyki mają widoczne na interfejsie parametry sterujące algorytmem próbkowania:

Automatyczny zrzut (czas): Ustawienie interwału czasowego analizy (na zrzucie: 6 sekund).

Prawdopodobieństwo zmiany: Zdefiniowanie progu czułości detekcji różnic (na zrzucie: wartość 40).

Możliwość precyzyjnej edycji tych wartości z poziomu UI pozwala na elastyczne dostosowanie systemu do dynamiki materiału wideo, eliminując konieczność ingerencji w kod źródłowy aplikacji przy zmianie zbioru badawczego.



Rysunek 5. Panel konfiguracyjny systemu z widocznymi ustawieniami algorytmu inteligentnego próbkowania (interwał 6s, próg zmiany 40%).

Warto również zwrócić uwagę na pole adresu źródłowego (widoczna na zrzucie ścieżka .../WizOdrec), które demonstruje elastyczność systemu w zarządzaniu danymi wejściowymi. Aplikacja została zaprojektowana tak, aby obsługiwać zarówno strumienie sieciowe (np. linki YouTube), jak i lokalne zbiory plików badawczych przechowywane na serwerze. Taka funkcjonalność była kluczowa dla realizacji badań porównawczych, umożliwiając płynne przełączanie się między standardowym materiałem wideo (Zbiór A) a przygotowanymi próbkami pisma odręcznego (Zbiór B) bez konieczności rekonfiguracji całego środowiska.

6. Wnioski

Przeprowadzone badania pozwoliły na zweryfikowanie hipotezy o zróżnicowanej skuteczności metod rozpoznawania tekstu w zależności od charakterystyki materiału źródłowego. Analiza wykazała, że nie istnieje jedno uniwersalne narzędzie optymalne dla wszystkich scenariuszy.

W przypadku pisma odręcznego i treści złożonych: Multimodalne modele językowe (w szczególności Gemini 1.5 Flash oraz Claude 3 Haiku) wykazały drastyczną przewagę nad tradycyjnymi algorytmami. Redukcja wartości metryki Levenshteina była w tych przypadkach wielokrotna względem lokalnych silników, co czyni modele AI jedynym skutecznym rozwiązaniem dla materiałów o niestandardowej strukturze, gdzie tradycyjny OCR (EasyOCR, Tesseract) generował wyniki nieużyteczne.

W przypadku standardowego tekstu cyfrowego (wideo): Tradycyjne rozwiązania (OCR.space, Tesseract) zachowały precyzję porównywalną z modelami językowymi, oferując jednocześnie znacznie krótszy czas przetwarzania. Dla materiałów o wysokim kontraście i standardowej czcionce użycie kosztownych obliczeniowo modeli LLM nie przynosi istotnej poprawy jakości.

Istotnym elementem badania była weryfikacja autorskiego mechanizmu inteligentnego próbkowania danych. Zastosowanie algorytmu detekcji zmian z progiem różnicy 40% pozwoliło na skuteczną eliminację klatek statycznych i duplikatów, redukując wolumen przetwarzanych danych do kluczowych próbek bez utraty informacji semantycznej. Potwierdza to, że wstępne przetwarzanie obrazu jest kluczowe dla wydajności systemów OCR.

Wyniki jednoznacznie wskazują na zasadność stosowania podejścia hybrydowego w systemach produkcyjnych. Optymalna architektura powinna wykorzystywać szybkie silniki lokalne do masowej analizy standardowych dokumentów, a zapytania do modeli multimodalnych kierować jedynie w przypadkach wykrycia pisma odręcznego lub konieczności głębokiej korekty błędów. Taka strategia pozwala zoptymalizować koszty operacyjne (tokeny API) przy zachowaniu maksymalnej skuteczności.

Zaimplementowany system demonstracyjny, oparty na Flask i MySQL, potwierdza techniczną wykonalność takiej integracji. Stanowi on fundament dla rozwiązań klasy enterprise, które w przyszłości będą coraz mocniej polegać na synergii klasycznych algorytmów przetwarzania obrazu z możliwościami rozumienia kontekstu oferowanymi przez Generatywną Sztuczną Inteligencję.

Literatura

1. Grinberg M., Flask. Tworzenie aplikacji internetowych w Pythonie, Helion.
2. Dokumentacja Flask - flask.palletsprojects.com/en/stable/
3. Dokumentacja OpenAI - platform.openai.com/docs/concepts/ dostęp źródła -2024
4. Dokumentacja Anthropic - docs.anthropic.com/ - 2024
5. Dokumentacja Gemini - ai.google.dev/gemini-api/ - 2024
6. Projekt Tesseract OCR na GitHub – github.com/tesseract-ocr/tesseract
7. Projekt EasyOCR na GitHub - github.com/JaidedAI/EasyOCR
8. OCR Space - narzędzie do rozpoznawania tekstu - ocr.space
9. Próbka badawcza na platformie youtube (wideo) - youtube.com/watch?v=fZsCg4mz364
10. Levenshtein distance - algorytm porównywania tekstów, en.wikipedia.org/wiki/Levenshtein_distance
11. Dokumentacja Python - docs.python.org/3/
12. Matplotlib biblioteka do generowania wykresów - matplotlib.org/3.5.3/index.html
13. Dokumentacja bazy danych MySQL - dev.mysql.com/doc
14. Artykuł branżowy - Gemini 1.5 – blog.google/intl/pl-pl/nowosci-produktowe/nasz-model-ai-nowej-generacji-gemini-15
15. Recenzja Claude - clickup.com/pl/blog/209690/recenzja-claude-ai