

## Narzędzie Analizy Podobieństwa Dokumentów

Automatyczne narzędzie do wykrywania wzorców powtarzającej się treści w dokumentach tekstowych (książki, artykuły, raporty, prace naukowe). Wykorzystuje zaawansowane techniki NLP do grupowania podobnych akapitów w klastry i generowania zwięzłych podsumowań dla każdej grupy.

### Funkcje

- **Automatyczna Analiza Treści:** Identyfikuje podobne akapity używając embeddingów zdań
- **Klasyfikacja Treści:** Kategoryzuje typy treści (spis treści, tabele, nagłówki, główna treść)
- **Podsumowania AI:** Generuje podsumowania klastrów używając modeli OpenAI GPT
- **Wiele Formatów Wyjściowych:** Eksport CSV, DOCX, JSON dla różnych przypadków użycia
- **Wieloplatformowość:** Działa na Windows i macOS
- **Wsparcie Dwujęzyczne:** Obsługuje treści polskie i angielskie z automatycznym wykrywaniem języka

### Szybki Start

#### Wymagania Wstępne

- Python 3.8 lub nowszy
- Klucz API OpenAI (opcjonalny, do podsumowań AI)

#### Instalacja

##### Opcja 1: Sklonuj repozytorium

1. Sklonuj repozytorium i przejdź do repozytorium:

```
bash

git clone https://github.com/your-username/document-similarity-analysis.git
cd document-similarity-analysis
```

##### Opcja 2: Użyj plików z pendrive

Jeśli używasz plików z pendrive np. D: to otwórz Terminal/PowerShell i przejdź na dysk/pendrive:

1. Przejdź na dysk/pendrive:

```
powershell

cd D:\
```

2. Sprawdź jakie pliki masz w tym katalogu:

```
powershell

dir *.txt
```




Jeśli widzisz plik requirements\_txt.txt zainstaluj zależności, jeśli nie znajdź plik z zależnościami.








#### Instalacja zależności

```
powershell

pip install -r requirements_txt.txt
```

#### Instalacja powinna zakończyć się zainstalowaniem wszystkich pakietów:





-  sentence-transformers (5.1.0) - do analizy podobieństwa dokumentów
-  scikit-learn (1.7.1) - narzędzia machine learning
-  numpy (2.3.2) - obliczenia numeryczne

-  python-docx (1.2.0) - obsługa plików Word
-  PyMuPDF (1.26.4) - obsługa plików PDF
-  pandas (2.3.2) - analiza danych
-  openai (1.106.1) - integracja z OpenAI
-  python-dotenv (1.1.1) - zmienne środowiskowe
-  torch (2.8.0) - PyTorch do deep learning
-  transformers (4.56.1) - modele transformerowe



### Konfiguracja klucza API (opcjonalnie)

Jeśli go nie masz lub go nie skonfigurujesz to aplikacja zadziała lokalnie.

#### Tryb offline (bez klucza API):

- Analiza podobieństwa dokumentów 
- Generowanie embeddingów 
- Klasteryzacja treści 
- Raporty CSV, JSON, DOCX 

#### Co wymaga klucza API:

- Automatyczne podsumowania klastrów 
- Analizy AI z analize\_results\_v3.py 

#### Aby skonfigurować klucz API:

1. Wejdź na stronę OpenAI: <https://platform.openai.com/api-keys>
2. Postępuj zgodnie z instrukcją, aby uzyskać swój klucz API

#### Opcja 1: Przez plik .env (zalecane)

```
powershell  
  
notepad "C:\Users\[TwojaNazwa]\Documents\DocumentAnalysis\.env"
```

Wklej:

```
OPENAI_API_KEY=twój_prawdziwy_klucz_tutaj
```

Zapisz plik (Ctrl+S) i zamknij.

#### Opcja 2: Przez zmienną środowiskową

```
powershell  
  
setx OPENAI_API_KEY "twój_prawdziwy_klucz_tutaj"
```

Następnie uruchom ponownie PowerShell i aplikację.

### Podstawowe Użycie

1. **Umieść swój dokument** w katalogu projektu, np.: `C:\Users\[TwojaNazwa]\Documents\DocumentAnalysis\Projects\[NazwaProjektu]`
2. **Uruchom analizę:**

```
bash  
  
python run_document.py --project moja_analiza --file "twój_dokument.pdf"
```

3. **Przejrzyj wyniki** w automatycznie utworzonym folderze projektu, np.: `C:\Users\[TwojaNazwa]\Documents\DocumentAnalysis\Projects\moja_analiza\`

### Testowanie Instalacji

Użyj dostarczonych plików testowych, aby sprawdzić czy wszystko działa:

```
bash

# Test z dokumentem angielskim
python run_document.py --project test_en --file "What is Lorem Ipsum.pdf"

# Test z dokumentem polskim
python run_document.py --project test_pl --file "Czym jest Lorem Ipsum.pdf"
```

### Kompletny Przepływ Pracy

```
bash

# Krok 1: Główna analiza
python run_document.py --project moja_praca --file praca.pdf

# Krok 2: Klasyfikacja treści
python analyze_results_v2.py --project moja_praca

# Krok 3: Podsumowania AI (wymaga klucza API)
python analyze_results_v3.py

# Krok 4: Wyodrębnienie klastrów
python extract_clusters.py

# Krok 5: Uzyskanie reprezentatywnych przykładów
python extract_examples.py --project moja_praca
```

### Pliki Wyjściowe

Plik	Opis
results_doc.csv	Surowe dane analizy z mapowaniem akapit-klastr
[nazwa_pliku]_analysis.docx	Raport analizy czytelny dla człowieka
[nazwa_pliku]_analysis.json	Eksport danych w formacie maszynowym
analysis_report.txt	Statystyki podsumowujące i spostrzeżenia
clusters/	Pojedyncze pliki klastrów

### Opcje Konfiguracji

#### Czułość Podobieństwa

- `--eps 0.3` (domyślnie) - Standardowa czułość
- `--eps 0.2` - Rygorystyczne dopasowanie (mniej, bardziej precyzyjne klastry)
- `--eps 0.4` - Luźne dopasowanie (więcej klastrów, szersze podobieństwo)

#### Minimalny Rozmiar Klastra

- `--min-samples 2` (domyślnie) - Uwzględnij wszystkie duplikaty
- `--min-samples 3` - Tylko klastry z 3+ akapitami

#### Wybór Modelu

- `--model gpt-4o` (domyślnie) - Najlepsza jakość, szczególnie dla polskiego
- `--model gpt-4o-mini` - Opcja budżetowa

### Obsługiwane Formaty Plików

- PDF** (.pdf) - Ekstrakcja tekstu z wykrywaniem akapitów
- Word** (.docx) - Natywna ekstrakcja akapitów
- Text** (.txt) - Wykrywanie akapitów przez podwójne łamanie linii

## Interpretacja Wyników

### Kategorie Podobieństwa

- **<10%**: Niskie podobieństwo - głównie unikalna treść
- **10-25%**: Pewne podobieństwo - powiązane tematy
- **25-50%**: Podobna treść - znaczące nakładanie się
- **50-75%**: Bardzo podobne - prawdopodobnie powtarzające się
- **>75%**: Niemal identyczne - silne powielanie

### Typy Treści

- **UNIQUE**: Pojedyncze akapity bez duplikatów
- **SIMILAR-XX**: Grupy powiązanej/powielonej treści
- **TOC**: Wpisy spisu treści
- **Header\_Footer**: Numery stron, nagłówki, stopki
- **Bibliography**: Referencje i cytowania
- **Table\_Figure**: Tabele, rysunki, studia przypadków

## Rozwiązywanie Problemów

### Częste Problemy

#### "Nie znaleziono treści"

- Sprawdź czy dokument zawiera czytelny tekst
- Spróbuj innego formatu pliku (PDF → DOCX)
- Upewnij się, że dokument nie jest oparty na obrazach

#### "Błąd importu"

- Zainstaluj brakujące zależności: `pip install -r requirements_txt.txt`
- Sprawdź wersję Pythona: `python --version` (potrzebne 3.8+)

#### "Nie znaleziono klastrów"

- Dokument może mieć bardzo unikalną treść (dobrze!)
- Spróbuj wyższej wartości eps: `--eps 0.4`
- Sprawdź minimalne próbki: `--min-samples 2`

### Błędy API

- Sprawdź czy klucz API jest poprawnie ustawiony
- Sprawdź kredyty na koncie OpenAI
- Narzędzie działa bez klucza API (bez podsumowań)

### Wskazówki Wydajnościowe

- Dla dużych dokumentów (> 100 stron) rozważ podział na rozdziały
- Użyj `gpt-4o-mini` do początkowych testów, aby zmniejszyć koszty
- Cache embeddingów jest automatycznie przechowywany dla powtarzanych analiz

### Struktura Projektu

```
twój_projekt/
├── src/           # Kod źródłowy
├── test_files/    # Przykładowe dokumenty do testowania
├── docs/          # Dokumentacja
├── requirements_txt.txt # Zależności Pythona
└── README.md      # Ten plik
```

Analiza tworzy foldery projektów w:

- **Windows:** C:\Users\[Nazwa]\Documents\DocumentAnalysis\Projects\
- **macOS:** ~/Documents/DocumentAnalysis/Projects/

### Przypadki Użycia

- **Prace Naukowe:** Znajdź powtarzające się wyjaśnienia lub definicje
- **Dokumentacja Techniczna:** Wykryj duplikujące się instrukcje
- **Raporty Biznesowe:** Zidentyfikuj nadmiarowe sekcje analizy
- **Książki i Artykuły:** Zlokalizuj powtarzające się tematy lub przykłady
- **Dokumenty Prawne:** Znajdź duplikujące się klauzule lub warunki

### Dodatkowe Zasoby

#### Konfiguracja dla Polskich Użytkowników

##### Konfiguracja Środowiska Windows:

```
cmd

# Ustawienie klucza API
setx OPENAI_API_KEY "sk-twój-klucz-tutaj"

# Sprawdzenie instalacji
python --version
pip list | findstr "openai\sentence"
```

##### Konfiguracja Środowiska macOS:

```
bash

# Dodaj do ~/.zshrc lub ~/.bash_profile
echo 'export OPENAI_API_KEY="sk-twój-klucz-tutaj"' >> ~/.zshrc

# Sprawdzenie instalacji
python3 --version
pip3 list | grep -E "openai\sentence"
```

### Przykładowe Projekty

#### Analiza Pracy Magisterskiej:

```
bash

python run_document.py --project praca_magisterska --file "praca_mgr.pdf" --eps 0.25
python analyze_results_v2.py --project praca_magisterska
python analyze_results_v3.py
```

#### Analiza Artykułu Naukowego:

```
bash

python run_document.py --project artykul_2024 --file "artykul.docx" --eps 0.3
python extract_examples.py --project artykul_2024 --examples 5
```

### Interpretacja Wyników dla Polskich Dokumentów

#### Optymalne Progi Podobieństwa:

- **Prace naukowe:** eps=0.25 (rygorystyczne wykrywanie)
- **Raporty biznesowe:** eps=0.35 (umiarkowane wykrywanie)
- **Artykuły:** eps=0.4 (szerokie wykrywanie tematyczne)

**Typowe Wzorce w Polskich Dokumentach:**

- Powtarzające się definicje w pracach akademickich
- Duplikujące się opisy metodologii
- Podobne wprowadzenia do rozdziałów
- Powtarzające się wnioski i rekomendacje

**Wkład w Projekt**

1. Zrób fork repozytorium
2. Utwórz swoją gałąź funkcji (`git checkout -b feature/NowaFunkcja`)
3. Commituj swoje zmiany (`git commit -m 'Dodaj jakąś NowaFunkcja'`)
4. Wypchnij do gałęzi (`git push origin feature/NowaFunkcja`)
5. Otwórz Pull Request

**Licencja**

Ten projekt jest licencjonowany na licencji MIT - zobacz plik [LICENSE](#) po szczegóły.

**Wsparcie**

- **Dokumentacja:** Zobacz `docs/business_case_v6_pl.md` dla szczegółowej dokumentacji w języku polskim
- **Problemy:** Utwórz issue na GitHubie dla błędów lub próśb o funkcje
- **Pytania:** Użyj GitHub Discussions dla pytań o użytkowanie

**Podziękowania**

- Zbudowane z [Sentence Transformers](#) do podobieństwa semantycznego
- Używa modeli [OpenAI GPT](#) do inteligentnego podsumowywania
- Klastrowanie DBSCAN przez [scikit-learn](#)