

Wyszukiwarka - Singular Value Decomposition zastosowanie

Michalina Hytrek

April 24, 2024

1 Cel

Stworzenie silnika wyszukiwarki wraz z aplikacją webową go wykorzystującą oraz pokazanie zastosowania Singular Value Decomposition. Dodatkowe założenie ograniczające wyszukiwania: w bazie mamy tylko pliki na temat Polski, wyszukiwanie jest dostępne tylko w języku angielskim.

2 Język oraz technologie

Do stworzenia projektu został użyty język programowania Python wraz z frameworkm Django. Cały kod (z wyjątkiem pobranych dokumentów oraz plikami z macierzami wynikowymi) znajduje się na [moim githubie](#).

3 Poszczególne kroki

1. Stworzenie/pobranie zbioru dokumentów
2. Określenie słownika słów kluczowych (bag of words)
3. Wyznaczenie wektora cech bag of words
4. Zbudowanie macierzy słowa x dokumenty
5. Przemnożenie macierzy przez "inverse document frequency"
6. Stworzenie modułu wczytującego zapytanie i wyliczenie miary prawdopodobieństwa
7. Usuwanie szumu i normalizacja macierzy

4 Tworzenie silnika wyszukiwarki

Cała implementacja dostępna na [githubie](#). Sam silnik znajduje się w folderze "engine".

4.1 Zbiór dokumentów

Wszystkie pobrane dokumenty tekstowe pochodzą ze strony [Wikipedia](#). Do ich pobrania użyto biblioteki pythonowej wikipedia oraz jej funkcji wyszukiwanie z wykorzystaniem tematu artykułu. Sciągnięte zostały dokumenty w tematyce:

- polityki
- historii
- kuchni
- sportu
- geografii
- kultury

Łączna ilość plików wynosi: 1116 (z każdego działu pobrane zostało około 200 artykułów)

4.2 Słownik słów i wektor cech dla dokumentów

Do stworzenia słownika słów wykorzystano wcześniej pobrane dokumenty zapisane jako obiekty, używając biblioteki Pickle. Po zebraniu listy unikatowych słów znajdujących się w dokumentach, poddano ją przefiltrowaniu (usunięto "stopwords", przyrostki z poszczególnych słów oraz zmieniono wszystkie litery na małe). Słownik słów wyniósł:

Wektor cech dokumentów został stworzony identycznie, ale z uwzględnieniem ilości wystąpień tematu w dokumencie oraz zamiast słów przetrzymujemy liczbę ich wystąpien w takiej samej kolejności jak w "bag of words".

Długość wektora bag of words:

4.3 Macierz wektorów cech

Macierz w której wektory cech ułożone są wektorowo (dokumenty reprezentowane są przez kolumny). W implementacji początkowo stworzona jest macierz transponowana, która następnie została ztransponowana. Następnie macierz została przemnożona przez "inverse document frequency":

$$IDF(w) = \log \frac{N}{n_w}$$

gdzie n_w jest liczbą dokumentów w których występuje słowo w , a N jest liczbą dokumentów. Kolejno macierz została znormalizowana.

4.4 Wprowadzanie zapytania i wyliczanie prawdopodobieństwa

Otrzymujemy od użytkownika zapytanie q , które traktujemy identycznie jak wektor cech dokumentów. Następnie wyliczamy miarę prawdopodobieństwa, do której używamy korelacji miedzy wektorami $/cos$:

1. Bez svd

$$\cos \theta = q \cdot T \cdot A$$

2. Z svd

$$\cos \theta = q \cdot U_k \cdot D_k \cdot V_k$$

5 Aplikacja

Strona została nazwana Poogle - od znanej wyszukiwarki i tematu informacji do wyszukiwania. Przy budowaniu aplikacji użyto framewoku Django oraz html z css. Strona wyszukuje zapytanie w około 1s. W przyszłości jest możliwa rozbudowa strony z wynikami o np. wypisywanie wraz z tytułem artykułu pierwszych kilku zdań. Aktualnie jedynie co dostajemy w wyniku wyszukiwania to tytuły artykułów wikipedii wraz z linkami.

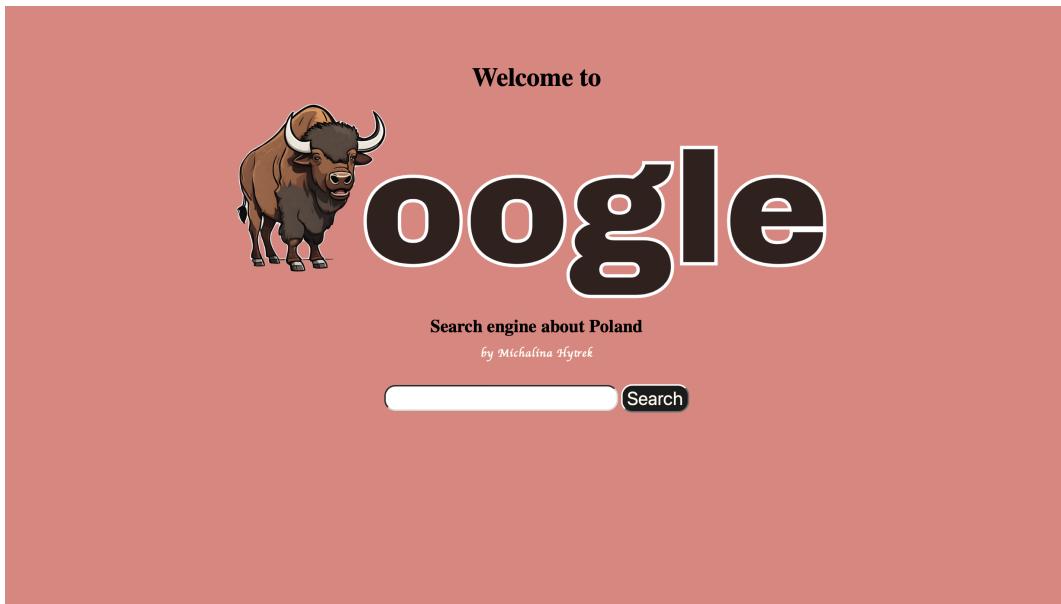


Figure 1: Strona główna aplikacji

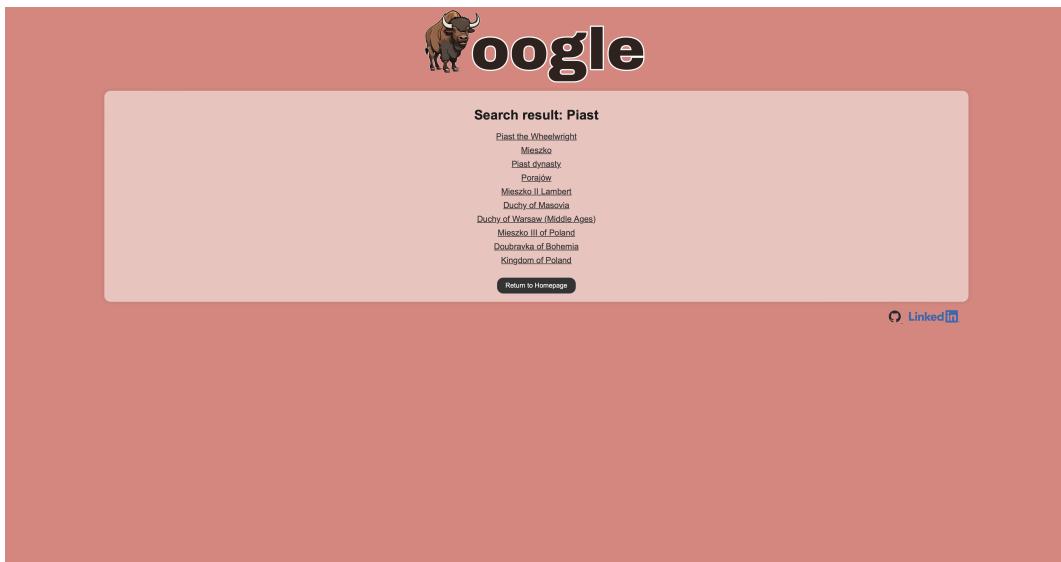


Figure 2: Strona z wynikami wyszukiwań

6 Porównanie wyników wyszukiwarki

Przeprowadzone testy:

6.1 Wynik dla zapytania: Dance

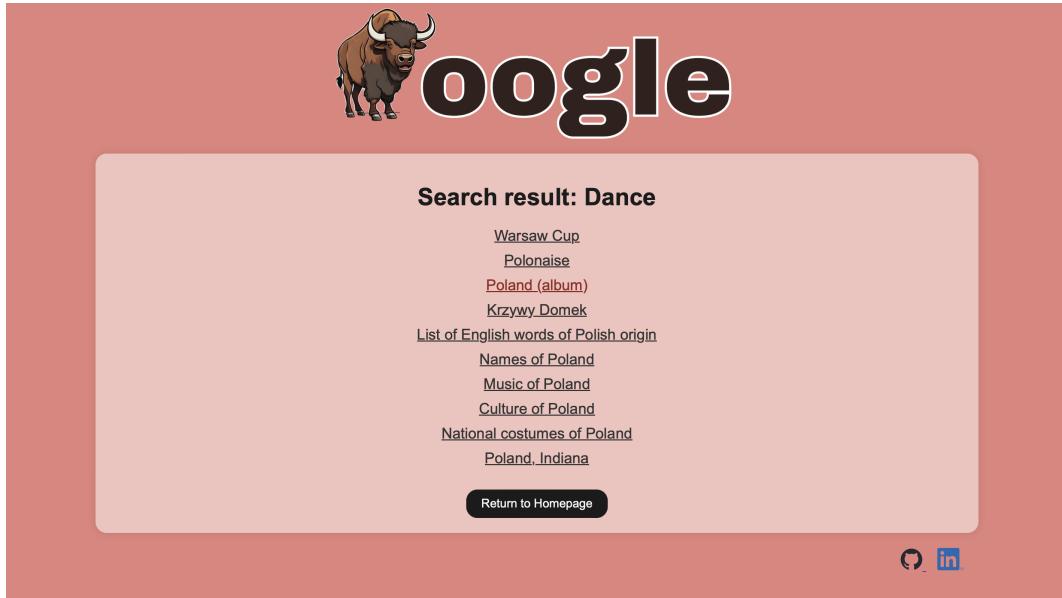


Figure 3: Wynik bez svd

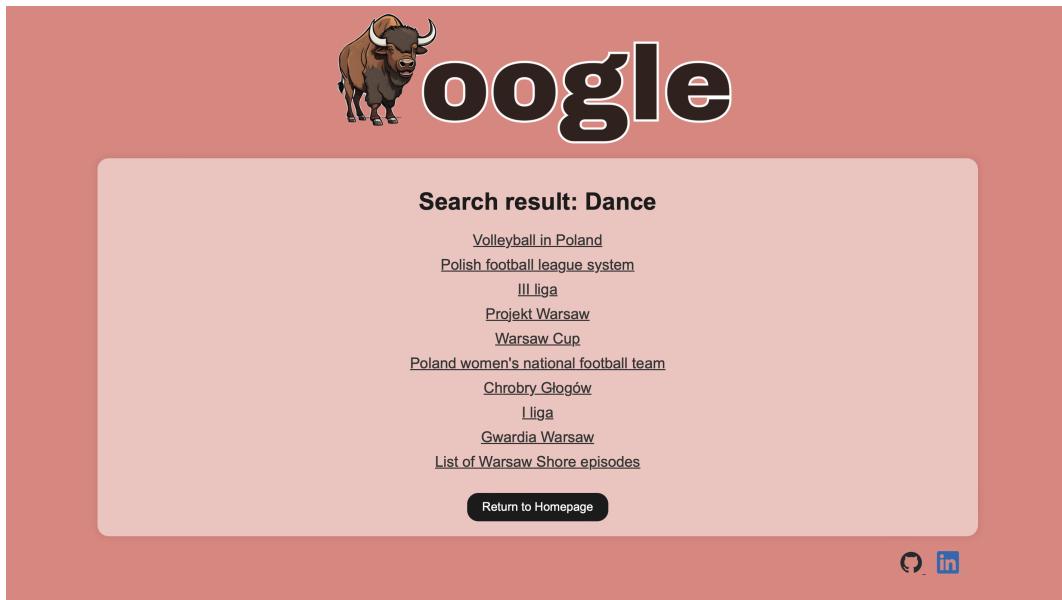


Figure 4: Wynik dla svd z k=10

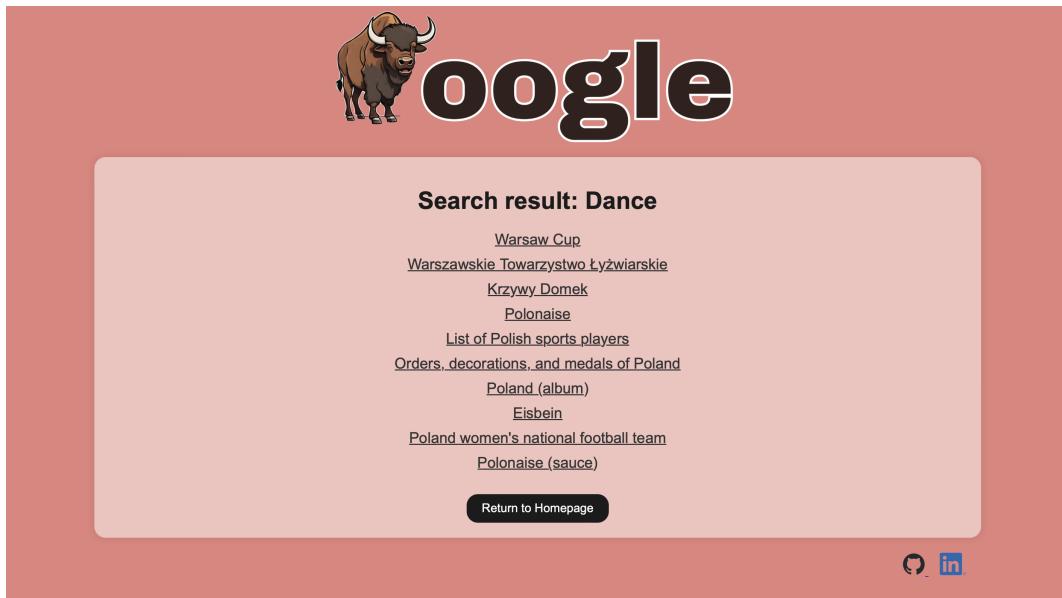


Figure 5: Wynik dla svd z k=100

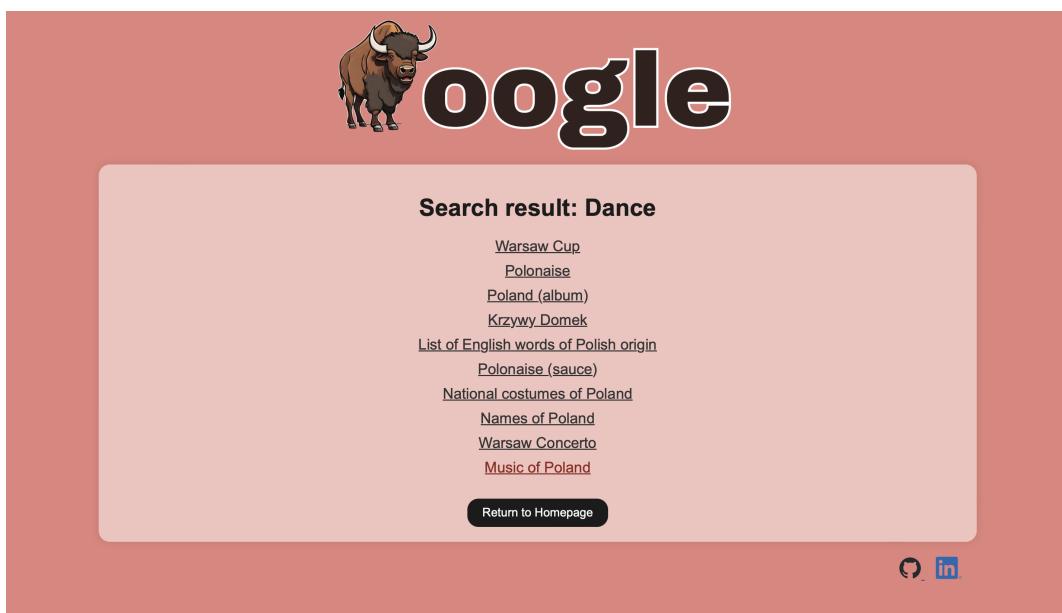


Figure 6: Wynik dla svd z k=500

6.2 Wyniki dla zapytania: Piast



Figure 7: Wynik bez svd



Figure 8: Wynik dla svd z k=10

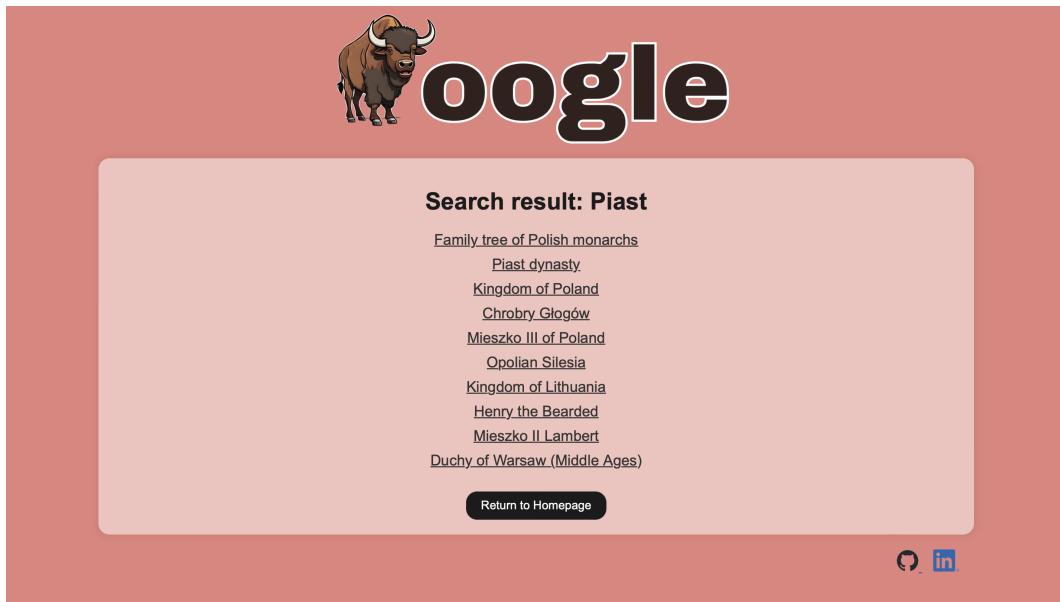


Figure 9: Wynik dla svd z k=100

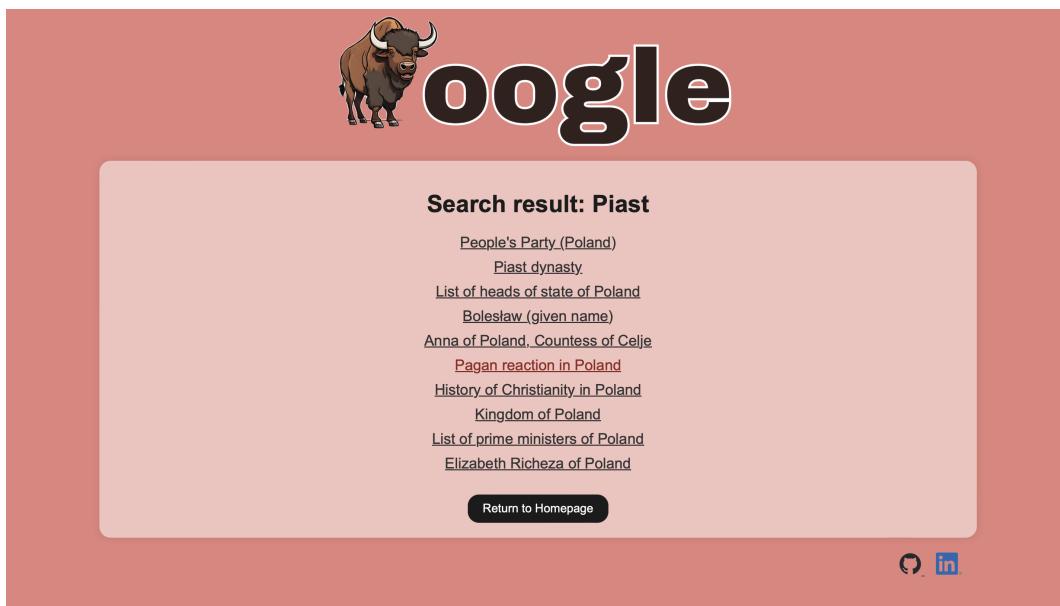


Figure 10: Wynik dla svd z k=500

6.3 Wyniki dla zapytania: The most popular sport



Figure 11: Wynik bez svd

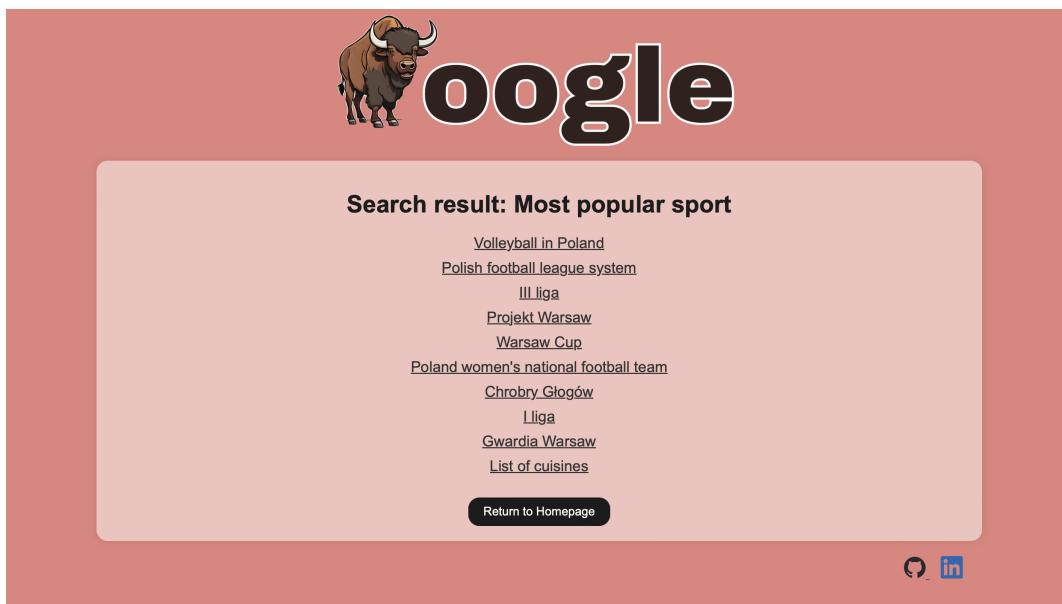


Figure 12: Wynik dla svd z k=10



Figure 13: Wynik dla svd z k=100

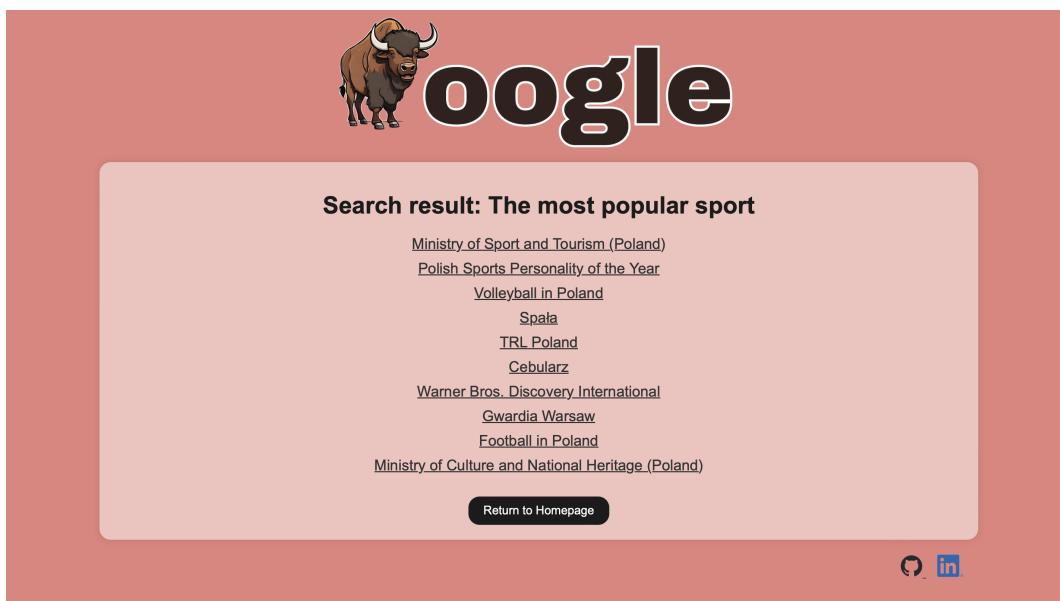


Figure 14: Wynik dla svd z k=500

6.4 Wyniki dla zapytania: War between Poland and Sweden in 16th century

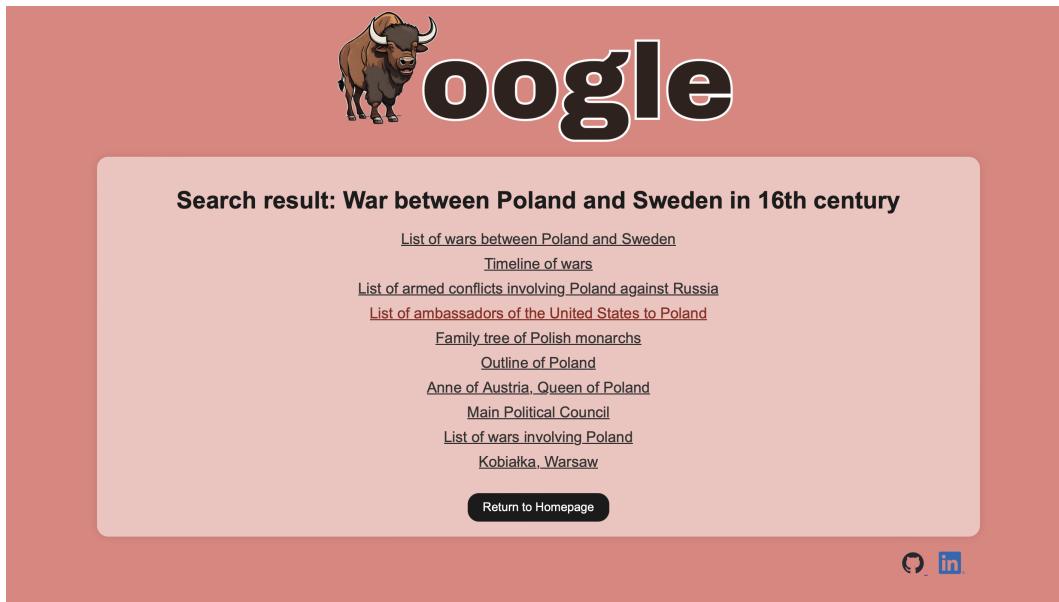


Figure 15: Wynik bez svd



Figure 16: Wynik dla svd z k=10

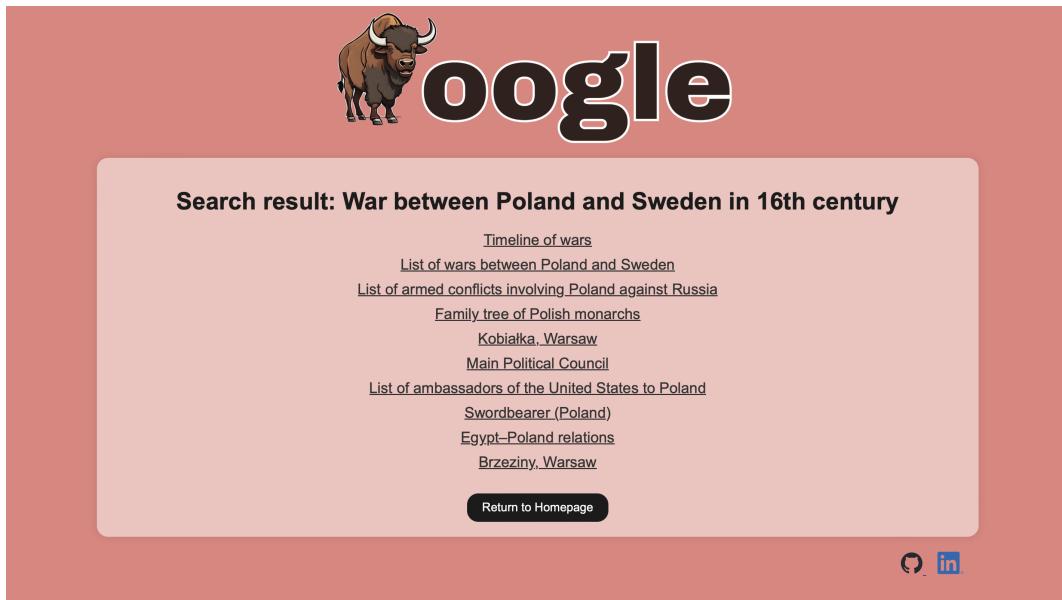


Figure 17: Wynik dla svd z k=100

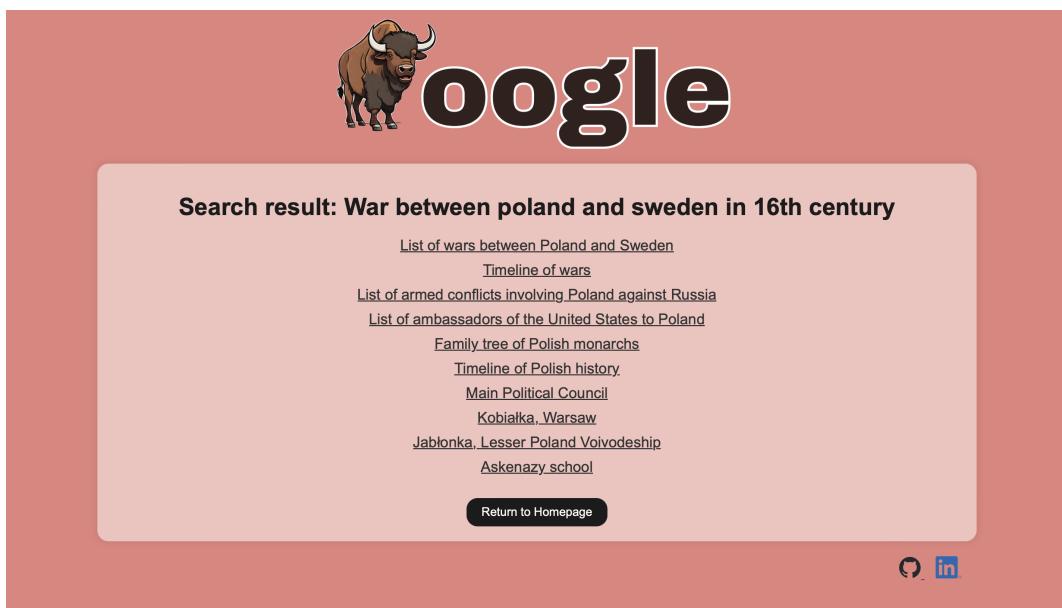


Figure 18: Wynik dla svd z k=500

6.5 Wyniki dla zapytania: 2024 European Parliament election



Figure 19: Wynik bez svd



Figure 20: Wynik dla svd z k=10



Figure 21: Wynik dla svd z k=100



Figure 22: Wynik dla svd z k=500

6.6 Wyniki dla zapytania: Dish of pork with breadcrumbs similar to schnitzel

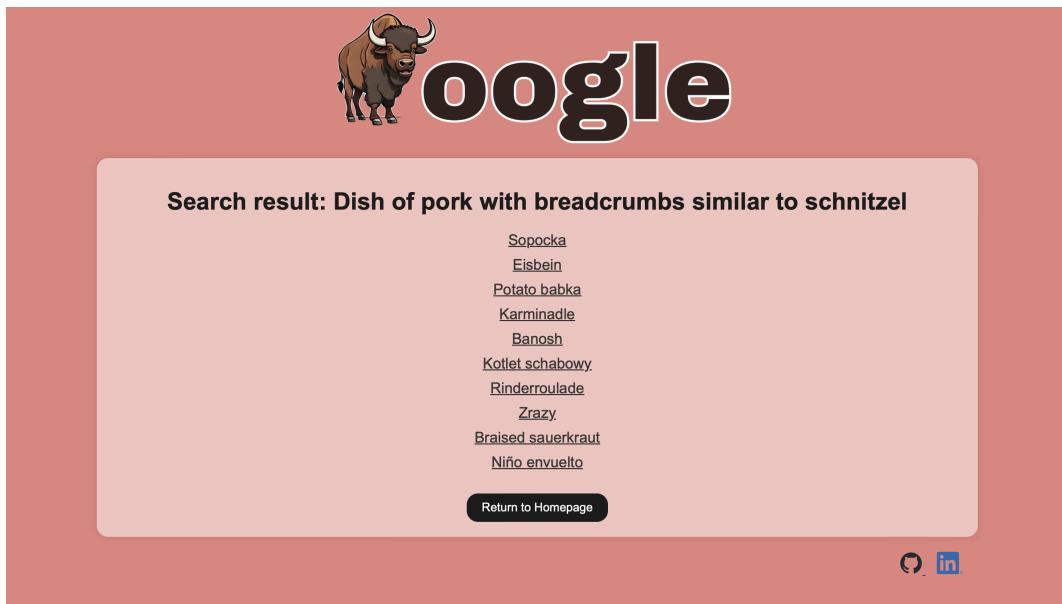


Figure 23: Wynik bez svd

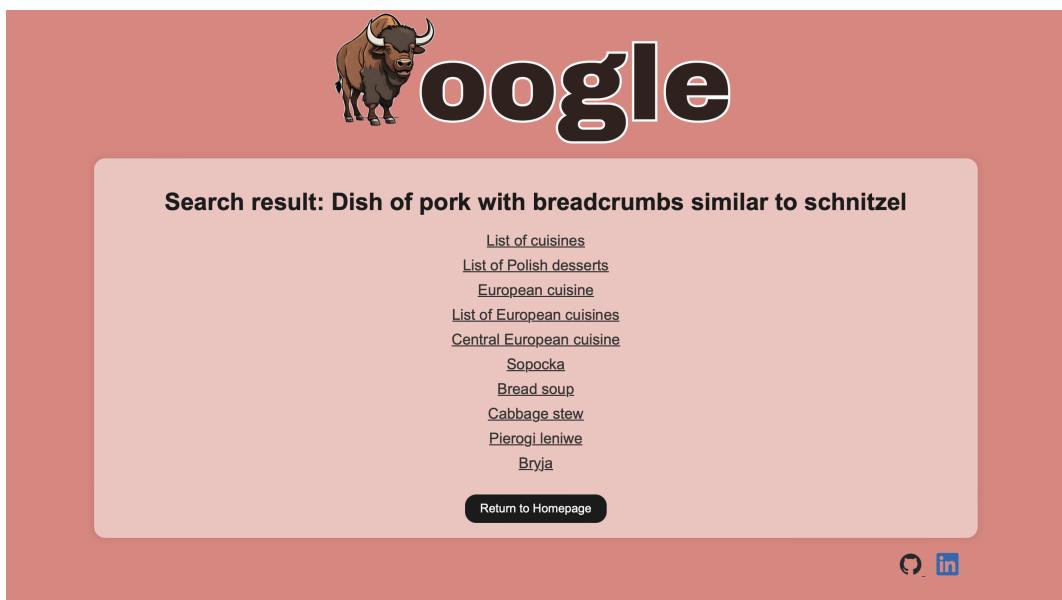


Figure 24: Wynik dla svd z k=10

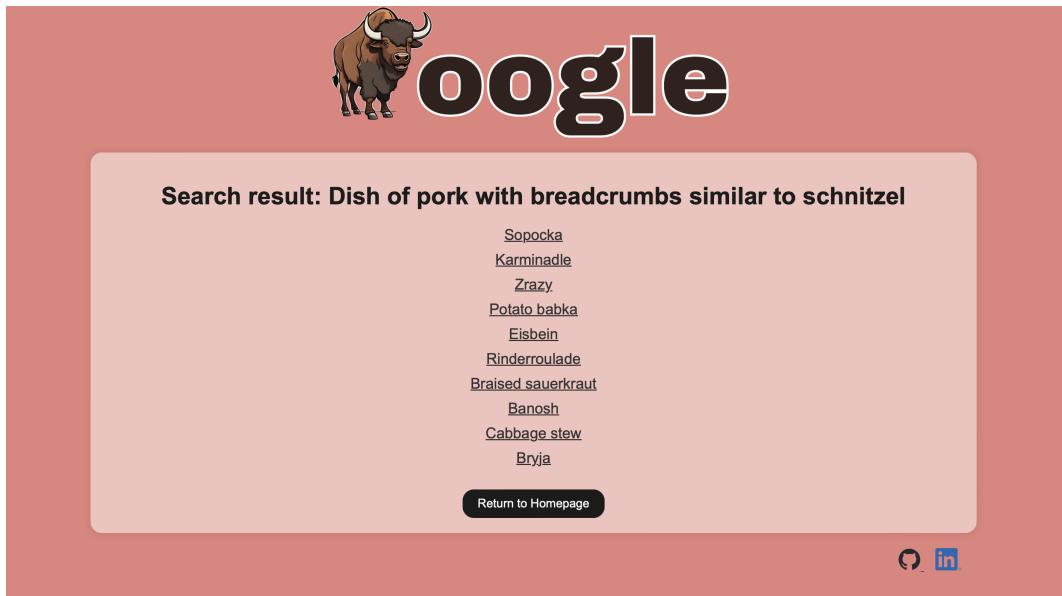


Figure 25: Wynik dla svd z k=100

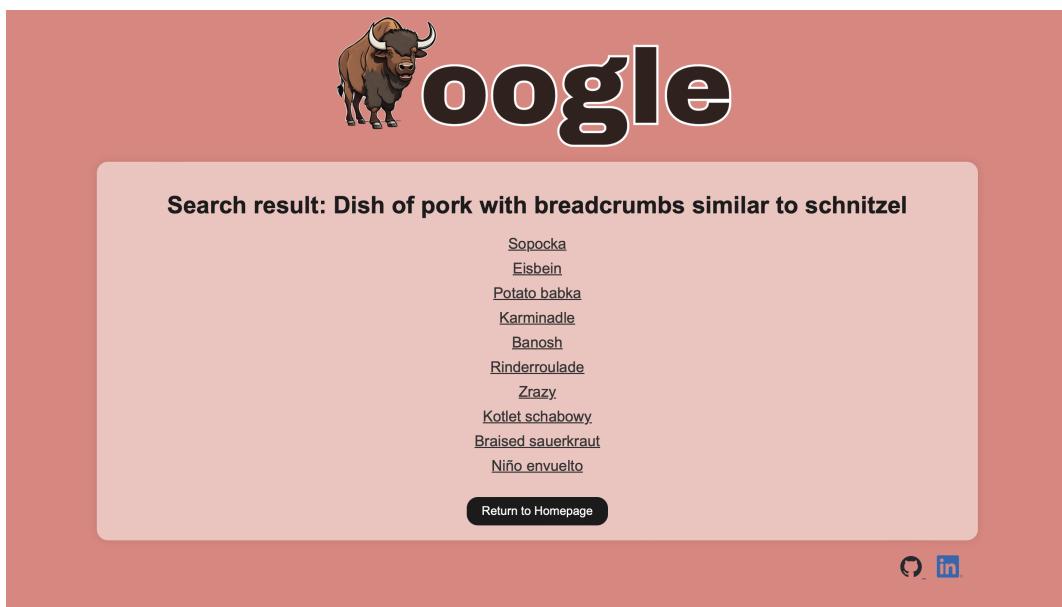


Figure 26: Wynik dla svd z k=500

6.7 Wyniki dla zapytania: Region in Poland with coal



Figure 27: Wynik bez svd

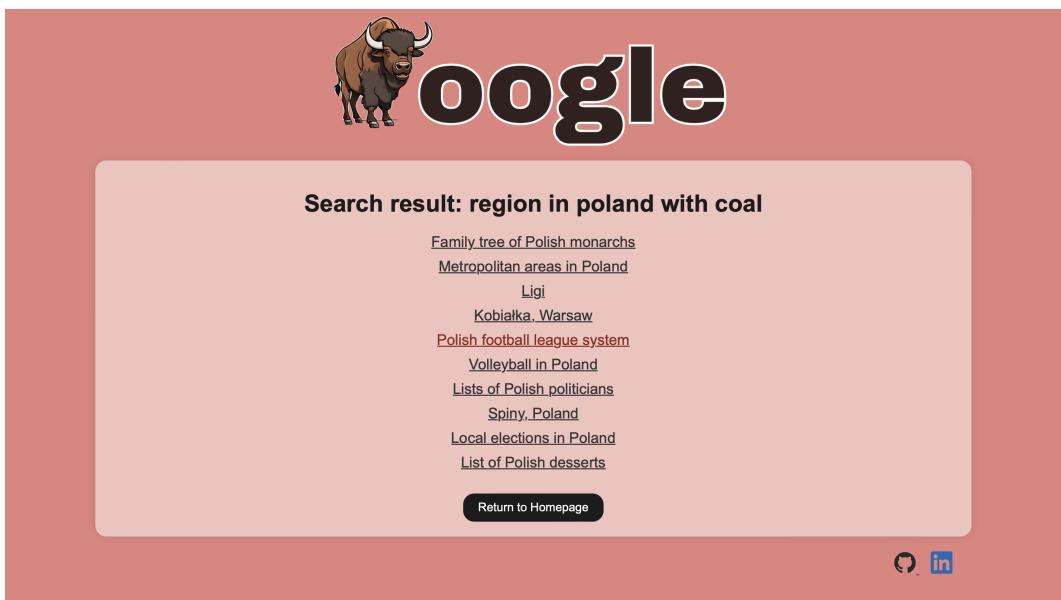
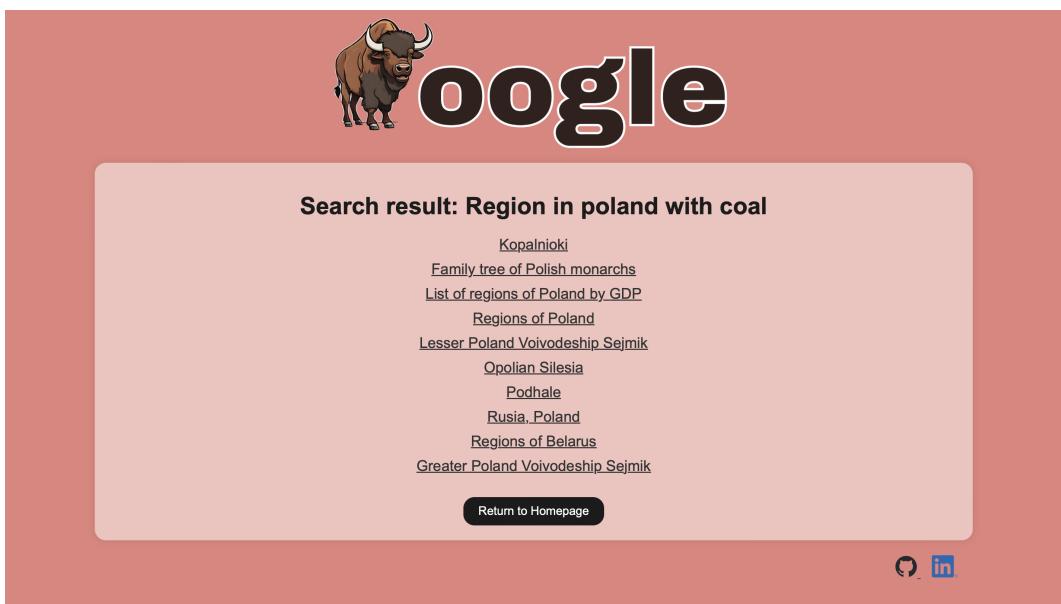


Figure 28: Wynik dla svd z k=10



© in

Figure 29: Wynik dla svd z k=100



© in

Figure 30: Wynik dla svd z k=500

7 Wnioski

Wyszukiwarka wypluwa artykuły o dobrej tematyce (związanej z zapytaniem). Daje zdecydowanie lepsze efekty przy bardziej unikatowych słowach takich jak: Piast, czy datach (przykład Figury 19,20,21,22).

Porównując odpowiedzi wyszukiwarki przy wyszukiwaniu z szumem i bez szumu, widać przewagę wyszukiwania z szumem (nie są to różnice). Jest on bardziej precyzyjny i lepiej dopasowuje artykuły do zapytania (przykłady Figury 7,8,9,10)

Następnie porównując wyszukiwanie bez szumu z stałą k=10, k=100, k=500,auważamy że najczęściej najlepsze efekty daje wyszukiwanie przy stałej k=500. Wyszukiwania przy innych stałych często dają odpowiedzi z artykułami które nie mają wiele wspólnego z naszym zapytaniem.

Podsumowując, wyszukiwarka lepiej radzi sobie z bardziej unikatowymi zapytaniami, a najlepsze efekty dawały wyzwukiwanie bez użycia svd.