Metody Obliczeniowe w nauce i technice

Laboratorium 4

Sprawozdanie

Mateusz Machowski

W celu przetestowania działania wyszukiwarki za zbiór danych przyjąłem około 2500 nagłówków artykułów z wikipedii. Pierwotnie chciałem użyć w zbioru tweetów, jednak okazało się że crawlowanie i wykorzystywanie api twittera jest dosyć problematyczne, więc postanowiłem wybrać artykuły. Są one przważnie dłuższe od tweetów, jednak dosyć znacząco krótsze od średniej długości artykułu.

Do wykonania zadania postanowiłem wykorzystać język Python w wersji 3. Python posiada wiele bibiliotek które znacząco ułatwiły wykonanie zadania:

Wikipedia – biblioteka pozwalająca na w prosty sposób zdobywanie treści artykułów i innych informacji o stronach wikipedii

Nltk – zbiór bibliotek pozwalających na operacje na tekstach, posiada kilka rodzajów stemmerów użytych w zadaniu oraz zbiory słów które można użyć do łatwego przeparsowania artykułów.

Numpy, scipy – klasyczne biblioteki pythonowskie do operacji na macierzach, scipy jest tutaj użyte głównie dla metod .sparse

Opis działania programu:

Webcrawler.py – pobiera nagłówki z losowych artykułów, zapisuje je w plikach oraz tworzy plik data który posiada historie sprawdzonych artykułów polączoną z nazwami plików

Search\_engine.py – skrypt posiadający wszystkie operacje związane z operacjami na macierzach, przygotowuje i zapisuje opracowane macierze A przed i po SVD, oraz termy wykorzystywane w wyszukiwaniu

Searcher.py – skrypt przyjmujący jako argumenty wyszukiwane słowa, wykonuje operacje korelacji za pomocą numpy, zwraca 10 najbardziej pasujących artykułów wraz z linkami.

Wyszukiwarka dla obydwu wersji macierzy działa zgodnie z przewidywaniami.

Wnioski:

- IDF ma znaczący wpływ na wyniki wyszukiwania. Docelowo chcemy aby termy które są rzadsze miały większy wpływ wyszukiwania niż te które występują we wszystkich artykułąch. Bez IDF możliwa jest sytuacja w której nie otrzymamy żadnego dobrego wyniku w wyszukiwaniu, bo jedno ze słów występuje we wszystkich artykułach. Dzięki IDF i nastawieniu na rzadkość wystąpień, elementy rzadsze są bardziej niż „premiowane” niż popularne, co zwiększa celność wyszukiwania.

- zbyt małe wartości k dla low rank approximation SVD powodują całkowite utracenie dobrych wyników wyszukiwania, niestety nie udało mi się zaobsewować różnic w działaniu wersji z szumami i bez szumów dla odpowiednio dobranych wartości k w przypadku najlepiej trafionych artykułów. Same wartości korelacji wektorów są mniejsze dla po usunięciu szumów dla najlepiej dobranych, jednakch ich kolejność jest mniej więcej taka sama.

Przykładowe wyszukiwanie dla wersji z szumami i bez szumów:

Query po stemmowaniu = ['macedonia', 'art', 'europe', 'embassy']

5 najlepszych wyników z szumami:

0.320363078811 https://en.wikipedia.org/wiki/Zivko\_Prendzov

0.223189429648 https://en.wikipedia.org/wiki/NSCAD\_conceptual\_art

0.161092430547 https://en.wikipedia.org/wiki/Libya%E2%80%93Malta\_relations

0.156082526451 https://en.wikipedia.org/wiki/Boredomresearch

0.142879069013 https://en.wikipedia.org/wiki/Clayton\_Brothers

5 najlepszych wyników bez szumu:

0.184417678995 https://en.wikipedia.org/wiki/NSCAD\_conceptual\_art

0.182623683046 https://en.wikipedia.org/wiki/Zivko\_Prendzov

0.172801545939 https://en.wikipedia.org/wiki/Kre\_M%27Baye

0.167817507678 https://en.wikipedia.org/wiki/Guillaume\_Bijl

0.16469347932 <https://en.wikipedia.org/wiki/Boredomresearch>

Jak widać dwa najelpsze wyniki są tylko zamienione kolejnością, jednak ich rożnica w korelacji jest znacznie większa w wersji z szumami.