Tobiasz Szulc

**Sprawozdanie 6**

**Wyszukiwarka**

1. **Web Crawling i Scraping**

Za web crawling i scraping odpowiedzialny jest program crawlerv2.py

Do wygenerowania zbioru dokumentów wykorzystałem biblioteki:

1. requests <- do wykonywania zapytań http
2. BeautifulSoup <- do wyłuskiwania linków z dokumentów html
3. Multiprocessing <- do przyśpieszenia działania programu

Program wczytuje poprzedni stan z plików

data/checkpoint\_dict.json i data/checkpoint\_set.json

Reprezentujące poprzedni stan (umożliwiając przerywanie i wznawianie działania programu)

Sieć jest przeszukiwana w sposób podobny do bfs, z wyjątkiem tego, że kolejność elementów w kolejce nie jest zachowywana pomiędzy uruchomieniami

Pobrane strony są zapisywane w odpowiadających plikach w folderze data/pages

2. **Przetwarzanie i generowanie macierzy**

Proces został podzielony na małe moduły „Taski”, każdy task ma własny plik źródłowy w folderze tasks, program matrix\_maker.py, który nadzoruje poprawną kolejność wykonania wszystkich zadań

Podstawowe biblioteki wykorzystane przy przetwarzaniu słów:

1. nltk <- word stemming, lista stop\_words, tokenizacja
2. BeautifulSoup <- wyłuskanie słów z dokumentu html
3. scipy / numpy <- efektywne operacje na macierzach (rzadkich)
4. Pickle <- przechowywanie danych pomiędzy taskami

Niektóre taski posiadają pliki konfiguracyjne w folderze config

Usuwanie słów / Zmniejszanie słownika następuje poprzez stemmowanie słów, usunięcie stopwords, słów zawierających znaki z poza zbioru liter ascii, posortowanie słów wg. Częstości występowania i pozostawienie ustalonej liczby słów ze środka (słowa występujace bardzo często lub bardzo rzadko są usuwane)

Program wytwarza macierze:

1. Macierz rzadką przed wliczeniem IDF, znormalizowaną
2. Macierz rzadką po wliczeniu IDF, znormalizowaną
3. Macierz po IDF i po wykonaniu SVD , wektorem normalizującym

3. **Search engine**

Program search engine występuje w dwóch wersjach:

Wersji consolowej (uruchomienie search\_engine.py)

Wersji gui (uruchomienie App.py)

Obie wersje wykorzystują wspólny kod odpowiedzialny za rozwiązywanie zapytań, wejście jest najpierw dzielone na słowa, stemmowane, wystąpienia znanych słów są wpisane do wektora, wektor jest normalizowany, i wymnażany z odpowiadającą macierzą (w zależności od trybu, IDF, NOIDF, SVD), w trybie SVD można wybrać ilość wektorów wykorzystanych w zapytaniu, następnie k stron z najwyższym wynikiem jest zwracana

Wersja gui wykorzystuje bibliotekę PyQt5

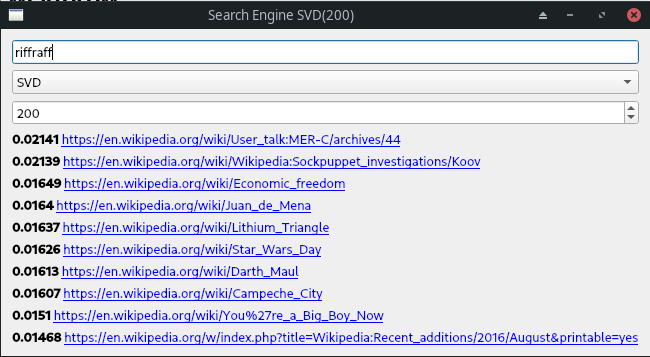
Można wykorzystać program get\_words.py w celu wygenerowania pliku tekstowego words.txt zawierającego wszystkie rozpoznawane przez program stemmy

4. **Porównanie działania na różnych macierzach**

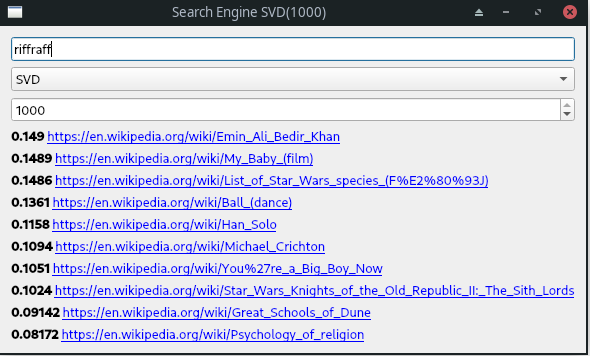
1. W przypadku wykorzystania macierzy bez IDF w wynikach strony, w których wyszukiwane słowa występowały mają bardzo wysokie oceny, wyniki w których słowa nie występowały otrzymywały automatycznie 0
2. W przypadku wykorzystania macierzy IDF w wynikach strony, w których wyszukiwane słowa występowały mają bardzo wysokie oceny, wyniki w których słowa nie występowały otrzymywały automatycznie 0, słowa, które występują na większej liczbie stron miały mniejszy wpływ na ocenę strony
3. W przypadku wykorzystania macierzy SVD wyniki stron były dużo niższe niż  w przypadku innych macierzy, ale wyniki niezerowe miały też strony, które nie zawierały żadnych z wyszukiwanych słów, zaczęły się pojawiać „grupy słów” występujących razem, gdzie wyszukanie jednego z nich było podobne do wyszukania wszystkich słów z grupy z pewną istotnością, stopień tego zjawiska można regulować ilością wykorzystanych wektorów

5. **Przykładowe użycie z różnymi parametrami**

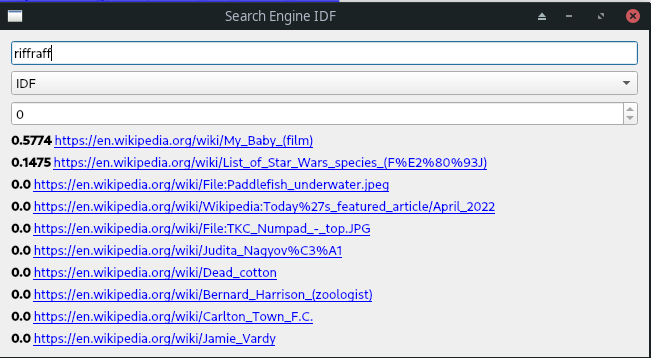
1. **SVD mała ilość wektorów**



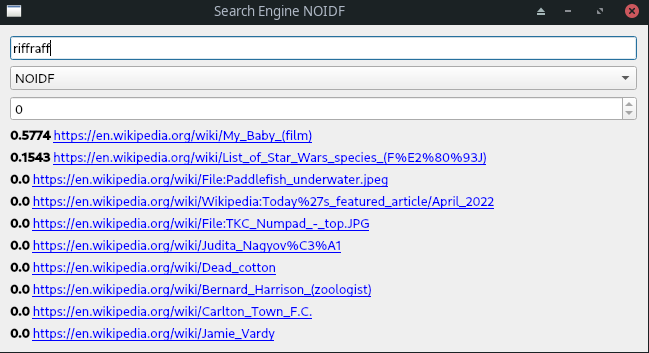
1. **SVD duża ilość wektorów**



1. **IDF**



1. **NOIDF**



1. **Wywołanie z linii poleceń**



6. **Informacje dodatkowe**

1. **Ze względu na wielkość macierzy nie zostaną one załączone z projektem**
2. **By zminimalizować czas odpowiedzi wyszukiwarki, wszystkie operacje, które mogły być wykonane przed wykonaniem zapytania zostały zawarte w macierzach wyjściowych, search\_engine wykonuje jedynie sprowadzenie zapytania do wektora i wymnożenie go z wybraną macierzą, przez co czasy odpowiedzi są praktycznie natychmiastowe**
3. **Ze względu na w.w. optymalizacje generowanie macierzy potrafi zajmować wiele godzin, jednakże są to obliczenia, które wystarczy wykonać tylko raz**