Singular Value Decomposition - zastosowania

Filip Piskorski[1]

[1]**Informatyka, WIEiT AGH**

ABSTRACT

W tym dokumencie przedstawię wyniki moich zmagań z SVD.

Keywords: svd, simple wikipedia, macierze

UŻYTE BIBLIOTEKI

Do reprezentacji i procesowania macierzy rzadkich używam biblioteki scipy z numpy. Pozwala ona wykonyważ operacje mnożenia macierzy, jak i również operacji na wektorach bag of words w macierzy A (np. dzielenie przez skalar każdego wektora odbywa się przez podzielenie przez wektor skalarów).
Oprócz tego wykorzystuję również bibliotekę pickle do zapisywania i wczytywania obiektów, multiprocessing, którą zrównoleglam obliczanie macierzy term-by-document i xml.etree.ElementTree do wyciągania elementów z pliku xml.

GŁÓWNE PROBLEMY I PRZESZKODY, Z KTÓRYMI SIĘ SPOTKAŁEM

Największym problemem jaki napotkałem podczas pisania tego projektu był rozmiar macierzy A po zastosowaniu svd. Przed przepuszczeniem jej przez svd macierz A waży około 30 MB, a po tej operacji waga jej rośnie do około 8 GB. Rozmiar jej zaczyna o tyle być problemem, że wraz z artykułami przestaje ona się mieścić w pamięci i ram, i karty graficznej na której akcelerowałem wcześniej obliczenia. Podejrzewam, że problem wynika z tego, że podczas mnożenia macierzy jest generowana pełna macierz gęsta. Niestety usunięcie zer z tej macierzy nie pomaga.

PREPROCESSING

preprocessing dokumentów

Czysty plik ściągnięty z wikipedii zawiera bardzo dużo śmieci. Na przykład: strony z edycji artykułów, na których ludzie ze sobą rozmawiają co z danym artykułem zrobić, kompletnie nic nie wnoszące artykuły po kilka liter, style itp.
Z tego powodu postanowiłem poczyścić ją trochę. Pousuwałem część stylów i komentarzy. Nie biorę też pod uwagę artykułów, które mają poniżej 1000 bajtów. Niestety nie udało mi się usunąć artykułów nienaukowych, które bardziej wyglądają jak fora. Rykoszetem też dostało się niektórym merytorycznym częściom artykułów.

preprocessing słów

Aby nie musieć mnożyć ogromnych macierzy ograniczyłem też zbiór słów, dla których tworzę worki słów. Najpierw usuwam wszystkie słowa poniżej 3 liter. Potem wycinam słowa od 43 do max_words, które mam ustawione na 10000 (ma to na celu redukcję wielkości macierzy, jak i usunięcie szumu wynikającego z losowych ciągów liter i mało wnoszęcych słów takich jak "the".)

preprocessing macierzy

Zgodnie z treścią zadania preprocesowałem też macierz dzieląc każdy wektor przez idf.

WPŁYW DZIELENIA PRZEZ IDF NA WYNIKI WYSZUKIWANIA

Subiektywnie uważam, że wpływ tej operacji jest pozytywny. Wyniki są trochę mniej losowe. Wydaje mi się, że wynika to z tego, że po tej operacji częste słow tracą na znaczeniu.

LOW RANK APPROXIMATION

Do przeprowadzenia low rank approximation wykorzystałem bibliotekę scipy.sparse.linalg.svds, która potrafi przeprowadzać lra dla podanego k. Niestety dla dużych k jest to bardzo czasochłonne ze względu na mnożenie macierzy w celu dostania macierzy po lra.

WPŁYW LRA NA WYNIKI WYSZUKIWANIA

Po przeprowadzeniu lra wyniki stają się jakkolwiek dobre dopiero przy k równym 3000. Przed tym wyniki wyszukiwania kompletnie nie mają związku z zapytaniem. Bardzo dobrze zaczyna się robić przy 9000 i przy tej wartości widać wpływ tej operacji na wyniki i usuwanie szumu.

CO WYSYŁAM WRAZ Z KODEM

Wraz z kodem wysyłam macierz po operacji idf, tytuły, teksty i przetworzone słowa wraz z częstotliwościami. Jest to wystarczająca ilość plików do uruchomienia aplikacji.

WNIOSKI

Moim głównym wnioskiem jest to, że lra może być użyte do usuwania szumów w macierzy i wydaje mi się, że może nawet przyspieszyć obliczenia. Niestety, mi nie udało się tego zrobić i u mnie obliczenia po aproksymacji macierzy są bardzo długie, a i sama macierz jest ogromna.