Metody Obliczeniowe w Nauce i Technice

Laboratorium 6 - Prosta wyszukiwarka internetowa
Sprawozdanie

Jakub Pinowski

2020-04-06

1 Wstęp

Do wykonania prostej wyszukiwarki internetowej użyty został język Python 3, dodatkowo, aby zapewnić bardzo prosty interfejs webowy, do podstawowego kodu dołączony został skrypt ładujący lokalny serwer Flask. Pobrane pliki tekstowe są losowymi artykułami z angielskiej Wikipedii, zawierają tylko tekst artykułu (maksymalnie 5000 znaków), bez metadanych.

2 Opis działania wyszukiwarki

Przed możliwością rozpoczęcia wyszukiwania potrzebny jest preprocessing podanych plików, który pozwoli na szybkie wyszukiwanie (przykładowo preprocessing dla około 50 tysięcy artykułów trwał 22 minuty).

Fazy preprocessingu:

- Wyznaczenie zbioru wszystkich termów w dokumentach
- Utworzenie mapy term $\rightarrow$ indeks i zapisanie jej do pliku
- Obliczenie macierzy *bag of words* (ilości występowania danego termu w dokumencie), macierz A
- Obliczenie *inverse document frequency* oraz zapisanie wyniku do pliku
- Podzielenie częstotliwości każdego słowa w macierzy przez jego *inverse document frequency*
- Znormalizowanie wektorów *bag of words* w macierzy i zapisanie powstałej macierzy do pliku
- Wykonanie dekompozycji SVD oraz zastosowanie *low rank approximation*
- Przy dekompozycji $U_k D_k V_k^T = A_k$, policzenie macierzy $D_k V_k^T$ i podzielenie każdej kolumny przez normę odpowiadającą tej kolumnie w macierzy A_k. Zapisanie do plików macierzy U_k oraz $D_k V_k^T$.

Trzeba dodać, że termami w rozumieniu naszej wyszukiwarki nie są wszystkie znalenione słowa dokumentu, najpierw słowo zostaje sformatowane przy użyciu *PorterStemmer* z modułu *nltk.stem.porter* (usuwanie prefiksów i suffiksów, takich jak *-ing, -s*). Spreparowane w ten sposób słowo musi spełniać wszystkie poniższe warunki:

- Jest poprawnym słowem języka angielskiego (*nltk.corpus.words.words()*)
- Nie jest wyrazem pomocniczym (the, at itp.), takie wyrazy zapewnia nam *nltk.corpus.stopwords.words("english")*
- Nie jest liczbą

Wyszukiwarka znajduje wyniki przy użyciu dwóch sposobów:

- Macierz *bag of words* ze znormalizowanymi kolumnami
- Sformatowana przez IDF oraz znormalizowana kolumnowo macierz *bag of words*
- Macierz po dekompozycji SVD i *low rank approximation* A_k

Oba sposoby rozpoczynają od policzenia poziomego wektora *bag or words* oznaczonego jako q oraz podzielenie jego elementów przez odpowiadający *inverse document frequency* i zastosowanie normalizacji wektora. Korelacja pomiędzy wektorem q jest już obliczana różnie, zależnie od sposobu wyszukiwania.

2.1 Sposób bez formatowania oraz po formatowaniu IDF

$$[cos(q, d_1), cos(q, d_2), ..., cos(q, d_n)] = qA$$

Dostajemy poprawne korelacje, ponieważ zarówno wektor q jak i kolumny macierzy A są znormalizowane.

2.2 SVD i *low rank approximation*

$$[cos(q, d_1), cos(q, d_2), ..., cos(q, d_n)] = qU_k D_k V_k^T$$

Tu również dostajemy poprawne korelacje, ponieważ zarówno wektor q jak i kolumny macierzy A_k (będące wynikiem działania $U_k D_k V_k^T$) są znormalizowane.

3 Wpływ zastosowania IDF

Zastosowanie *inverse document frequency* wynika z faktu, iż pewne słowa są bardziej powszechne w ogólnym użyciu niż inne. Z tego względu chcemy zmniejszyć wpływ słów często występujących w wektorach *bag of words*, tak aby artykuły były rozpoznawalne wśród innych głównie ze względu na słowa specyficzne dla tego dokumentu.

Przykład: zapytanie *society screw* Do zapyrania użyte zostało słowo stosunkowo często pojawiające się w artykułach *society* oraz słowo rzadko pojawiające się *screw*. Oba sposoby wysukiwania pokazały nam na liście wyników artykuły **Twin-screw steamer** oraz **Nanded City, Pune**. Pierwszy bardzo krótki posiada w swojej treści słowo *screw* 5 razy, słowo *society* 0 razy. Drugi nieznacznie dłuższy ma 0 wystąpień *screw* 13 wystąpień *society*.

	Twin-screw steamer	**Nanded City, Pune**
Bez formatowania	0.2786	**0.3840**
Po zastosowaniu IDF	**0.5045**	0.3765

Oczywiście w drugim tekście szukane słowo występuje gęściej niż w tekście pierwszym, przez co wyszukiwarka bez użycia IDF faworyzuje tekst **Nanded City, Pune**. Po zastosowaniu formatowania okazuje się jednak, że występowanie słowa *screw* jest dużo bardziej nietypowe, dzięki temu korelacja wektora dla dokumentu **Twin-screw steamer** z wektorem naszego zapytania jest prawie dwukrotnie większa niż przed nim.

4 Porówanie wyników przed i po zastosowaniu SVD i *low rank approximation*

Problemem wyszukiwania w macierzy wektorów *bag of words* jest dosłowność interpretacji zapytania, próbując odnaleźć jakieś słowa kluczowe będziemy mnożyć rzadki wektor q i rzadkie kolumny macierzy A, w wielu przypadkach odpowiadające komórki w macierzy będą równe 0, ponieważ słowa może w tekście nie być.

W przypadku wyszukiwania przy pomocy macierzy A_k sytuacja się nieco zmienia, ponieważ w wyniku *low rank approximation* wartości w wektorach *bag of words* termów występujących często obok siebie wpływają na siebie, powodując że wartość w wektorze dla termu, który nie występuje w dokumentcie może być znacząca, jeśli w tym dokumencie występuje słowo, które jest do niego podobne semantycznie (w rozumieniu matematycznym, a nie dosłownym znaczeniowym).

Przykład: zapytanie *sportsman*

Artykułem o drugiej najwyższej korelacji w wyszukiwaniu SVD jest strona o tytule **Vladimir Kuznetsov**. Prezentuje ona listę osób o tym imieniu i nazwisku, nie ma tam ani jednego wystąpienia słowa *sportsman*, jednak widzimy słowa takie jak *olympic, player, footballer, athlete, cyclist*. Widzimy tutaj naturalne powiązanie słowa kluczowego z zawartością artykułu mimo braku bezpośredniego powiązania oraz nie definiując wcześniej powiązań między termami.

Nie zawsze jednak działa to tak jak byśmy chcieli. Czasami wyniki są powiązane z naszymi słowami kluczowymi, jednak zbyt odbiegają od tego czego oczekujemy.

Przykład: zapytanie *cancer treatment*

Wyszukiwanie przy pomocy macierzy A jako jeden z najbliższych wyników daje artykuł **ECHO-7**, który dość dosłownie mówi właśnie o leczeniu raka. Wyniki uzyskane dla macierzy A_k są mocno powiązane z tematem medycyny, słowa takie jak *disease, condition, clinic, syndrome*. Są one jednak mocno odległe od zadanej tematyki. Może to być spowodowane faktem, że znalezione artykuły były bardzo krótkie, więc takie słowa pojawiają się tam gęsto. Nie są to jednak wyniki jakich byśmy oczekiwali.

Widzimy więc, że zastosowanie *low rank approximation* pozwala uzyskać nietrywialne wyniki wyszukiwań, nawet gdy żadne reguły powiązania pewnych grup wyrazów nie zostały zdefiniowane. Nie jest to jednak narzędzie niezawodne i bazuje na prostych regułach matematycznych, a wyniki tych działań nie są w żaden sposób weryfikowane w poszukiwaniu błędnych powiązań. Nie zmienia to faktu, że jest to duży krok w stronę dobrej wyszukiwarki, szczególnie że *low rank approximation* pozwala nam znacznie skompresować dane jakich potrzebujemy do wyszukiwania. Dużo zależy od wyboru liczby składowych jakie weźmiemy pod uwagę przy tworzeniu macierzy A_k. Zbyt duża wartość k powoduje, że wyniki stają się bardzo dosłowne i nie różnią się wiele od tych uzyskanych dla A. Z drugiej zbyt mała wartość daje wyniki oderwane całkowicie od tego co chcemy otrzymać. Wartością dla której uzyskiwane rezulaty były najbardziej celne uznałem *ilość_termów/50* i dla takiej wartości zostały wykonane podane wyszukiwania.

5 Infomacje na temat implementacji wyszukiwarki i interfejsu webowego

Do uruchomienia silnika wyszukiwarki potrzbny jest lokalny interpreter języka Python z zainstalowanymi pakietami

- *numpy*
- *scipy*
- *nltk*

Kod odpowiedzialny za silnik wyszukiwarki znajduje się w *src/s_engine.py*

Dodatkowo do uruchomienia serwera lokalnego potrzebne są pakiety

- *flask*
- *flask-wtf*

Pliki tekstowe, które zostaną użyte przez wyszukiwarkę muszą znajdować się pod ścieżką *resources/articles* od katalogu w którym uruchamiany jest serwer. W folderze tym znajduje się tylko 1000 przykładowych plików z artykułami, ponieważ 50 tysięcy takich artykułów zajmuje około 150 MB miejsca na dysku.

Serwer może zostać szybko uruchomiony przy użyciu skryptu *run_server.sh*