Metody Obliczeniowe w Nauce i Technice

Labolatorium 8

Singular Value Decomposition - zastosowania

Mateusz Praski

Informatyka Rok 2
AGH WIET
20 V 2020

Spis treści

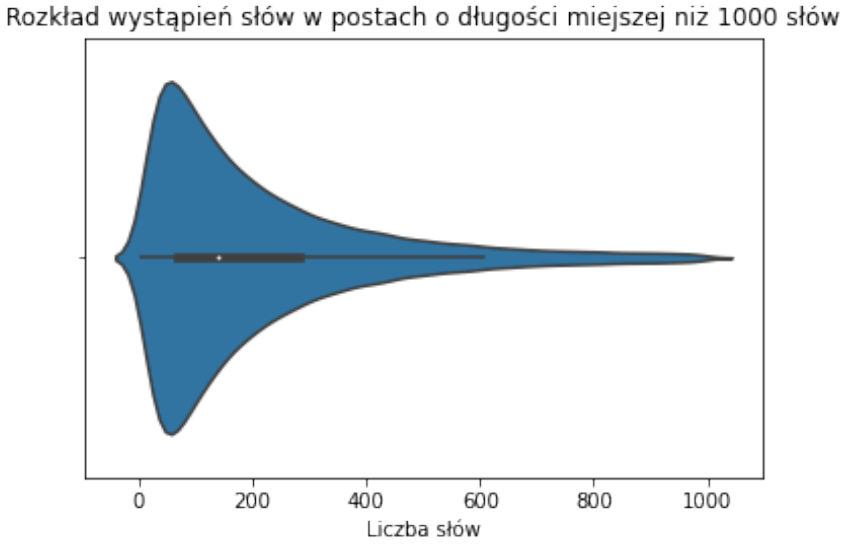
1 Przygotowywanie zbioru danych

1.1 Zebranie danych

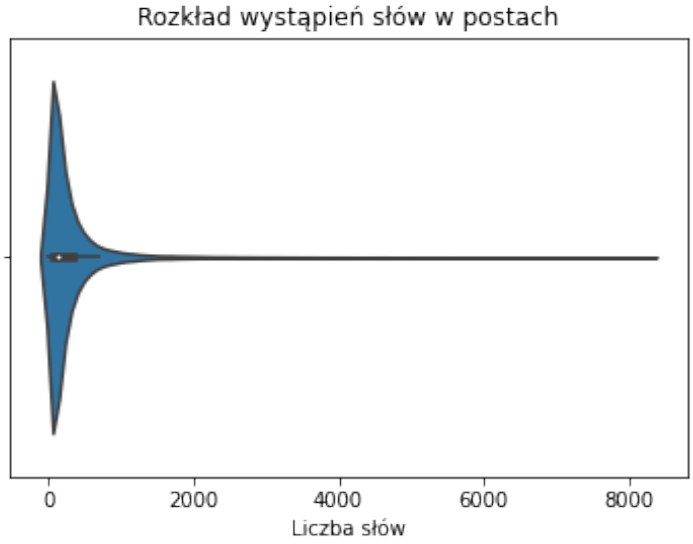
Jako źródło danych został wybrany serwis internetowy *Reddit.com*. Udostępnia on API pozwalające na zbieranie danych tekstowych ze strony w prosty sposób. Ponadto dla języka Python dostępny jest pakiet PRAW (Python Reddit API Wrapper), ułatwiający cały proces.

Jako zbiór Subredditów z których zbierano dane wybrano ponad 500 najpopularniejszych, a z każdego z nich zebrano blisko 500. Posty bez opisu zostawały odfiltrowane, a *scraper* działał do osiągnięcia progu rozmiaru bazy danych, czyli 65 001 postów z opisami. Dla każdego postu zebrano takie dane jak tytuł, opis, nazwa subreddita, link (niestety okazało się później, że nie wszystkie poprawnie odnoszą do postu) oraz inne informacje.

Jako dokumenty przyjęto konkatenację tytułu, działu oraz treści posta (wszystkie elementy są równie ważne). Średnia długość dokumentu wyniosła 300 ± 497 słów. Poniżej znajdują się odpowiednie wykresy wiolinowe rozkładów rozmiaru dokumentów.



Rysunek 1: Wykres wiolinowy liczby słów w dokumentach dla dokumentów krótszych niż 1000 słów



Rysunek 2: Wykres wiolinowy liczby słów w dokumentach

1.2 Preparacja danych

Posty zostały poddane wstępnej obróbce początkowej. Składały się na nią:

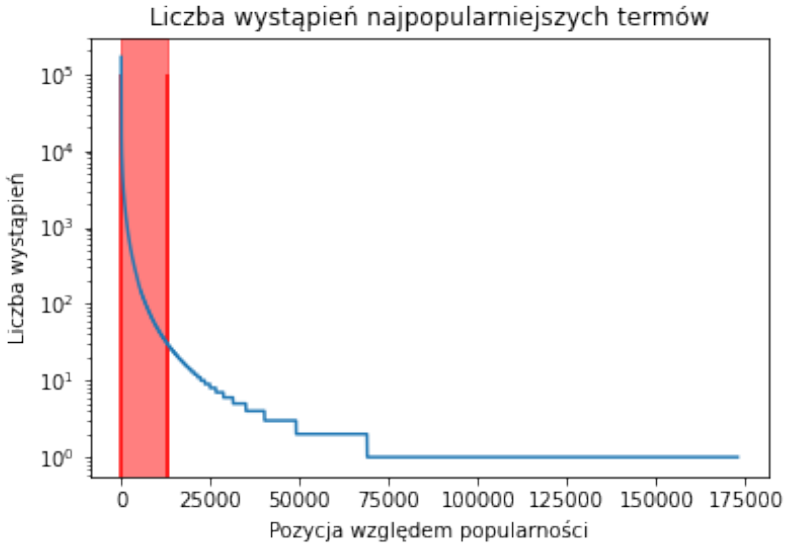
- Zamiana emoji na opisy w ASCII,
- Usunięcie znaków interpunkcyjnych (!"$%\'()*+,-./:;<=>?@[]_‘{|}~)
- Usunięcie cyfr,
- Usunięcie znaków poza zakresu ASCII,
- Usunięcie *stop wordów* (wykorzystana została tutaj lista z biblioteki *nltk* dla języka angielskiego),
- *Stemming* tekstu.

2 Wektoryzacja tekstu

2.1 Bag-of-words

Następnym etapem w procesie przygotowania indeksacji dla wyszukiwarki było stworzenie *bag-of-words* przestrzeni wektorowej. Do tego celu wykorzystano zbiór wyrazów znajdujących się w indeksowanym zbiorze. Pominięte zostały wyrazy występujące rzadziej niż 30 razy we wszystkich dokumentach (dolne 92.5% termów) oraz wyrazy występujące częściej niż 30 000 razy (górne 0.02%). Pozwoliło to na zmniejszenie rozmiaru przestrzeni termów z 172 881 do 12 824. Powodami usunięcia tak znaczącej części słownika były między innymi:

- Ponieważ Reddit jest forem internetowym, istnieje prawdopodobieństwo literówek przez użytkowników, które przez algorytm byłby traktowane jako nowe słowa,
- Wysoko wymiarowe przestrzenie wpływają negatywnie na jakość analizy (*curse of dimensionality*),
- Mimo redukcji rozmiaru słownika do 10% stanu oryginalnego 95.8% termów w dokumentach zostało zachowanych



Rysunek 3: Wykres popularności termów wraz z zaznaczonym obszarem wybranego *bag-of-words*

2.2 Wektoryzacja

Do indeksacji z wykorzystaniem *inverse document frequency* została stworzona kolejna mapa z indkesu termu na jego wartość IDF w postaci:

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

Gdzie N jest liczbą dokumentów, a n_w liczbą dokumentów zawierającą term w.

Na podstawie powyższych powstały dwie macierze rzadkie. Pierwszą z nich jest macierz konstruowana na podstawie liczby wystąpień słowa w poście. Druga z nich jest macierzą korzystają z wartości IDF do przydzielana współrzędnych wektora. Obie macierze powstały z wykorzystaniem znormalizowanych wektorów dokumentów, dzięki czemu miara podobieństwa na etapie wyszukiwania nie musi być normalizowana. Jako miara podobieństwa dokumentu d_j do zapytania q zostało przyjęte podobieństwo cosinusowe opisane poniższym wzorem:

$$(q^T A)^T = [\cos\theta_1, \cos\theta_2, ..., \cos\theta_n]$$
$$sim(q, d_j) = \cos\theta_j = (q^T A)[j]$$

Gdzie $X[j]$ jest j-tą pozycją wektora.

3 Reprezentacja wyników

Do wprowadzania wyszukiwań do przetworzonej bazy danych została napisana aplikacja webowa oparta na *Bootstrapie* oraz *Django*. Aplikacja pozwala na wybranie metody wyszukiwania spośród macierzy z liczbą wystąpień, IDF oraz *low rank approximation*. W przypadku ostatniego aplikacja pozwala wybrać stopień aproksymacji w przedziale od 1 do rozmiaru załadowanej dekompozycji SVD. Możliwe jest również wybranie liczby postów w wyniku zapytania.

Dla każdego posta w wyniku zapytania zostają zwrócone tytuł, link, dział oraz pierwsze 100 znaków postu. Do uruchomienia aplikacja potrzebuje następujących plików

- smalldb.csv - plik csv wczytywany przez pandas, z tytułami, opisami, linkami i działami postów,

- *map.pickle* - mapa termów na indeksy wektora,
- *count.npz* - znormalizowana macierz rzadka *term-by-document* z liczbą wystąpień słów,
- *svd.pickle* - krotka (U, S, V^T) dekompozycji SVD macierzy z liczbą wystąpień słów (opcjonalnie do wyszukiwania z *low rank approximation*)
- *svd_idf.pickle* - krotka (U, S, V^T) dekompozycji SVD macierzy IDF (opcjonalnie do wyszukiwania z *low rank approximation*)
- *idf.npz* - znormalizowana macierz rzadka *term-by-document* z IDF (opcjonalnie do wyszukiwania z IDF)
- *idf_map.pickle* - mapa indeksów termów na wartości IDF (opcjonalnie do wyszukiwania z IDF)

Aplikacja automatycznie wczyta pliki z podanego katalogu. Katalog można skonfigurować zmieniając zmienną *DIR* w pliku *searchengine/catalog/params.py*.

4 Implementacja low rank approximation

Dla macierzy częstościowej została obliczona dekompozycja SVD pierwszych 1 500 wartości własnych macierzy *term-by-document* i zapisana do plików *svd.pickle* oraz *svd_idf.pickle*. Z powodu rozmiaru macierzy, niemożliwa była jej normalizacja przed zapisaniem. Dlatego dla zapytań korzystających z SVD dokonywana jest następująca normalizacja:

$$
\begin{aligned}
A_k &= U_k D_k V_k^T \\
U_k &= [u_1|...|u_k] \\
S_k &= D_k V_k^T \\
s_j &= S_k e_j \\
sim(q, d_j) &= \frac{q^T U_k s_j}{||s_j||}
\end{aligned}
$$

W ten sposób jesteśmy w stanie optymalnie znormalizować wyniki wyszukiwania korzystając z Numpy. Ponieważ liczenie normy dla wszystkich wektorów zajmuje znaczącą część czasu wyszukiwania, aplikacja zapisuje do cache'a wcześniej obliczone współczynniki normalizacji.

5 Wyniki

Silnik znajduje z powodzeniem odpowiedzi na proste zapytania długości 1-3 słów. Dla dłuższych oraz złożonych zapytań wyniki są mniej zadowalające. Główną przyczyną tego jest wykorzystanie unigramów słownych, które nie pozwalają na wiązanie słów w zapytaniu.

Nie zauważono znaczących różnic pomiędzy wynikami dla IDF oraz zliczania dla zwykłych zapytań. Kolejności wyników różnią się, jednakże w większości przypadków zbiory pokrywały się.

5.1 Low rank approximation

Low rank approximation dało bardzo intrygujące wyniki. Bez żadnego wsparcia od strony „ludzkiej" algorytm na postawie dekompozycji SVD potrafił znajdować posty, które nie zawierały bezpośrednio poszukiwanych słów, jednakże poruszały tematy podobne (w kontekście postów na reddicie) dla przykładu:

- Dla wyszukiwań nt. „historii rpg" SVD znajdowało posty na temat gier fabularnych, podczas gdy bezpośrednie zapytania zwracały posty nt. pisania opowieści,
- Dla zapytania „tales" SVD zwracało posty nt. opowieści, historii które nie zawierały słowa kluczowego,

- Dla zapytania „bitcoin” SVD odnajdowało posty nt. innych kryptowalut, które nie wspominały o BTC,
- Dla zapytania „unicorn” wyniki wyszukiwania jawnego nie mają nic wspólnego ze sobą, poza wyszukiwaną frazą. Wyszukiwanie przez SVD skojarzyło tę frazę z kryptowalutą Uniswap.

SVD z IDF w jednych przypadkach sprawdzało się lepiej od zliczania, a w innych gorzej. Jako przykład dla zapytania „bitcoin” *Count SVD* znajdowało posty o innych kryptowalutach, gdy *IDF SVD* głównie o BTC. W innym przypadku dla zapytania „tales” zwykłe zliczanie uznawało tematykę zapytania za związaną z pisarstwem oraz grami, podczas gdy IDF łączył je z fantastyką.

Dla *low rank approximation* testowane wartości rangi były w zakresie od 100 do 1500. Najlepiej sprawdzały się wartości w przedziale 350 - 500. Na nich algorytm był w stanie zauważyć rzeczy wymienione wyżej. W przypadku wyższych wartości (np. 1000, 1200) wyniki były już bardzo podobne do tych zwracanych przez zwykłe zapytania. Dla wartości niższych (np. 100, 200) w większości przypadków nie było widać kontekstu stojącego za wynikami wyszukiwania.

5.2 Porównanie ze zbiorem bez optymalizacji *bag-of-words*

Dla zbioru niezoptymalizowane posiadającego ponad 170 000 termów, *latent semantic indexing* nie był w stanie osiągnąć interesujących wyników. Były one zbyt słabo związane ze sobą (r = 500, 700), bądź zbyt podobne do standardowych metod (r = 900, 1200).

6 Uwagi co do plików załączonych do sprawozdania

Z powodu rozmiaru plików potrzebnych do uruchomienia bazy danych zostały one umieszczone na dysku Google pod następującym linkiem:

`https://drive.google.com/drive/folders/13rNG55UObS_7iCNAh9AHKFB_sc4vgxWK?usp=sharing`

Literatura

[1] Carl D. Meyer *Matrix Analysis and Applied Linear Algebra*
`http://www.cse.zju.edu.cn/eclass/attachments/2015-10/01-1446085870-145420.pdf`

[2] Wenhu Chen, Yu Su, Yilin Shen, Zhiyu Chen, Xifeng Yan, William Wang *How Large a Vocabulary Does Text Classification Need?A Variational Approach to Vocabulary Selection*
`https://arxiv.org/pdf/1902.10339.pdf`