

Metody Obliczeniowe w Nauce i Technice

Laboratorium 6

Adam Woźny

Zadanie 1 - Wyszukiwarka

1 – proces tworzenia struktury umożliwiającej wyszukiwanie

1.1 – wstępne przetwarzanie i tokenizacja tekstu

Wyszukiwarka artykułów pochodzących z uproszczonej Wikipedii, pobranych już jako wstępnie przetworzony dokument tekstowy z serwisu Github. Zawiera on 50411 teksty w formacie tytuł \n treść, oddzielonych pustymi liniami. Program po kolei wczytuje każdy artykuł wraz z tytułem, następnie dzieli treść na tokeny (pojedyncze słowa oraz znaki interpunkcyjne), dokonuje selekcji, oraz stemuje każdy z tokenów za pomocą algorytmu Porter Stemmer. Na podstawie tak przetworzonych tokenów tworzony jest słownik słów kluczowych (należą do niego wszystkie przetworzone tokeny powstałe podczas wstępnego przetwarzania tekstu) w którym każdemu słowu przyporządkowana jest unikalna najmniejsza liczba porządkowa.

```
def __create_tokens_dict(self):
    file = open(self._file_name, encoding = 'utf8')
    text = file.read()
    tokens = set(self._transformator.transform(text))
    tokens_dict = {token: nr for nr, token in enumerate(tokens)}
    return tokens_dict
```

1.2 – tworzenie macierzy wystąpień (bag of words)

Na podstawie wyżej wymienionego słownika dla każdego artykułu tworzony jest pionowy wektor o rozmiarze (rozmiar_słownika x 1) w którym dana współrzędna wektora oznacza ilość tokenów w tekście o tym numerze w słowniku. Połączone w ten sposób wektory tworzą macierz o wymiarach (rozmiar słownika x liczba artykułów). Jak można się domyślić jest to macierz rzadka gdzie ponad 99% jej wartości to 0. Z powodów technicznych ta macierz została przedstawiona jako obiekt `scipy.sparse.lil_matrix`, bo tylko w tej formie byłam w stanie zmieścić ją w pamięci RAM

```
def __create_matrix(self):
    matrix =
scipy.sparse.lil_matrix((len(self._tokens_dict), self.number_of_articles))
    stemmer = nltk.stem.PorterStemmer()
    gen = DataLoader('corpus.txt')
    stop_words = set(nltk.corpus.stopwords.words('english'))
    for i in range(self.number_of_articles):
        tokens = nltk.tokenize.WordPunctTokenizer().tokenize(gen.next())
        list = [stemmer.stem(token) for token in tokens if len(token) >= 3
and token.lower() not in stop_words ]
        counter = Counter(list)
        for word in counter:
            matrix[self._tokens_dict[word], i] = counter[word]
    self.contents = gen.contents
    return matrix
```

1.3 – przetwarzanie macierzy wystąpień

Wiersze wyżej wymienionej macierzy zostały przemnożone przez współczynnik IDF oraz później znormalizowane. Mnożenie każdego wiersza przez element wektora IDF technicznie zostało

rozwiązanie za pomocą mnożenia transponowanej macierzy przez wektor. W procesie normalizacji korzystałem z funkcji bibliotecznej, która dzieli każdy element wektora przez jego długość, dając na wyjściu wektor jednostkowy.

1.4 – usuwanie szumu z macierzy za pomocą SVD

Do tak przetworzonej macierzy zastosowałem metodę przybliżenia za pomocą macierzy niższego rzędu (low rank aproximation). Pozwoliło to uzyskać efekt rozpoznawanie kontekstu słów oraz o wiele szybsze wyszukiwanie. Technicznie wykorzystałem do tego funkcję TruncatedSVD, która zwraca przybliżenie macierzy za pomocą jej pierwszych k rzędów, według wartości osobliwych. W ten sposób powstaje nam macierz o wymiarach ilość_artykułów $\times k$.

2 – prezentacja wyszukiwarki

Wyszukiwarka po wprowadzeniu danego hasła (może to być wyraz lub ich zbiór), prezentuje artykuły wraz z współczynnikiem dopasowywania w kolejności według wcześniej wspomnianego współczynnika.

2.1 – przykładowe wyniki dla zapytań

Animal - ('Animal', 0.7128976303751962), ('Anime', 0.6994570156435314), ('Animation', 0.612836011210292), ('Animalia', 0.609932891018067), ('Animal Planet', 0.5930529409200129), ('Animal rights', 0.5697055699678943), ('Animator', 0.5374261619361617), ('Biogeography', 0.5334609696606876)

Tutaj jak widać z racji na bardzo popularne słowo wyszukiwania zostały dopasowane ze względu na pisownię (n.p wyraz Anime nie ma nic wspólnego jeśli w sensie tematycznym z zapytaniem)

Month - ('August', 0.12526672906915973), ('February', 0.11754587398537522), ('January', 0.11050877412567534), ('Month', 0.0960096908369665), ('Hebrew calendar', 0.08895460698699373), ('December', 0.08741158373407952), ('October', 0.08672200187390203), ('November', 0.08617691144698542), ('September', 0.08312779108610535), ('New Year's Day', 0.08070296199090027), ('January 1', 0.08027777990313151), ('July', 0.07940105335036962), ('Common year', 0.07776176540551741), ('May', 0.07416946291597624)

W tym przypadku z kolei kiedy zapytanie występuje rzadziej, widać rozpoznawanie kontekstu artykułu, słowo „month” występuje w wyżej wymienionych artykułach od 1 do 5 razy, jak również w wielu innych, ale dzięki redukcji szumów SVD wyszukiwarka ma możliwość szukania z udziałem kontekstu, co dało bardzo poprawne wyniki zgodne z tematem zapytania, a nie tylko z składnią słowa.

Autobus - ('Public transport', 0.010805541843491133), ('United States Department of Transportation', 0.010139808848102703), ('Autobus', 0.009907225874538274), ('Transport for London', 0.009796943928492589), ('Rail transport', 0.008690336249762617), ('Transport', 0.008645724787648749), ('United States Secretary of Transportation', 0.00819572465904454), ('Baby transport', 0.007596205174718801), ('Transportation in Pakistan', 0.007428640530140188), ('Mary Peters (politician)', 0.006914075759054512), ('Brihanmumbai Electric Supply and Transport', 0.005888351839947813)

W tym przypadku potwierdza się hipoteza, która mówi, że jeśli zapytanie występuje bardzo rzadko bezpośrednio w artykułach to bardzo łatwo zaobserwować wyszukiwanie za pomocą kontekstu. We wszystkich artykułach poza pierwszym i trzecim nigdzie nie występuje słowo „autobus”, a jednak zostało one skojarzone z tematyką transportu publicznego i do tego skojarzenia zostały dobrane wyniki.


Rock music - ('Rock and roll', 0.6961148152981915), ('Art rock', 0.6912819683426608), ('Pop music', 0.6260742737450307), ('Rock music', 0.5892774503882948), ('C-rock', 0.5793617291981035), ('Pop rock', 0.578497239403959), ('Rock band', 0.5766647794543606), ('Rock', 0.5652758977784053), ('Progressive rock', 0.5432585353452173), ('Indie (music)', 0.5364726917694953), ('Industrial rock', 0.5328669430229722), ('Music', 0.5299065852027848), ('Alternative rock', 0.509066826512399), ('Christian rock', 0.4996157344632174), ('Sedimentary rock', 0.4833268090755052),

W tym przypadku można zauważyć poprawną analizę kolokacji słów, ponieważ „rock music” to gatunek muzyczny, co zostało zauważone pomimo tego, że tylko jeden z tych wyrazów ma podobny kontekst, a drugi całkiem odrębny. Widać również, że gdy słowa kluczowe zapytania często występują w artykułach to efekt analizy kontekstu nie jest tak bardzo widoczny.

Ecology - ('Models of nature', 0.048546926945092266), ('Ecology', 0.04418374491440166), ('Nature', 0.04101740498174579), ('Nature's services', 0.04079010017049913), ('Natural environment', 0.04057628578119803), ('Robert Costanza', 0.039946043580040926), ('Political economy', 0.03776648405140094), ('Land (economics)', 0.03506143758919537), ('Natural capital', 0.03410770710314642), ('Environment', 0.033712499980044455), ('Natural gas vehicle (NGV)', 0.03234581427028268), ('Ecoregion', 0.029176104885764033), ('Ecological yield', 0.027925559137454137), ('Geographer', 0.02715142047983369), ('Ecosystem valuation', 0.026610069874523322), ('Returns (economics)', 0.026564223866927025),

W tym przypadku widać, że jeśli słowo kluczowe występuje bardzo rzadko to może zająć efekt zbyt szerokiego kontekstu. Na przykładzie wyników związanych z ekologią widać, że kontekst tego wyrazu był zbyt szeroki.

2.2- prezentacja aplikacji

 Browser

— □ ×

SZUKACZ

Wpisz zapytanie

Wyszukaj

Strona główna wyszukiwarki pozwalająca na wprowadzenie zapytanie

WYNIKI

Wyniki wyszukiwania dla zapytania "animal"

1. Współczynnik dopasowania: 0.713

Animal

Animals are living things. Animals are not plants so they can't make their own food or energy by themselves. Animals have to eat other living things (animals, plants, fungi, etc.) to get energy to live. Many animals live in this world. Some are big and some are small. Some live in water, others live on the ground and some animals can fly. Being able to move from one place to another is another distinctive characteristic of an animal. Some animals eat only plants; they are called herbivores. Other animals eat only meat and are called carnivores. Animals that eat both plants and meat are called omnivores. Animals are divided into groups; see animalia. Their mode of nutrition is known as heterotrophic nutrition because they eat other living organisms as food in order to survive.

There are animals living in solitary and groups. Examples of animals living in solitary are tigers, rhinoceros, cheetahs and more. Examples of animals living in groups are coyotes, bees, monkeys and more. Animals living in solitary do not have to share food with each other. Animals living in groups have better protection against becoming the prey of others. Bees and ants are grouped in a special group called a colony. In a colony, the animals work together in a special way and usually have a leader. The leader of the bees is called the Queen Bee.

The animal kingdom is very diverse. There are many types of animals. But, the common animals you know are only about 3% of the animal kingdom. There are many other animals, such as insects and sponges! Animals can mainly be divided into two main groups, the invertebrates, and the vertebrates. The vertebrates are animals with a backbone, or spine, and the invertebrates without. Invertebrates include insects, crustaceans, molluscs, corals, worms, and much more. The vertebrates are divided into 5 groups, fish, birds, amphibians, reptiles and mammals.

[Previous](#)[Next](#)

Strona z wynikami, przyciski next oraz previous pozwalają na przeglądanie artykułów zgodnie z współczynnikiem dopasowania.

Bibliografia:

- Marix Analysis and Applied Linear Algebra, Carl D. Mayer, SIAM, 2000.
- <https://docs.scipy.org/doc/scipy/reference/sparse.html>
- https://scikit-learn.org/stable/user_guide.html
- <https://doc.qt.io/qtforpython/>
- <https://www.hindawi.com/journals/sp/2019/1095643/>
- <https://nlp.stanford.edu/IR-book/html/htmledition/latent-semantic-indexing-1.html>
- <https://numpy.org/doc/stable/user/index.html>
- <https://www.nltk.org/>
- <https://www.youtube.com/watch?v=nODp9ter2Og>
- <https://www.sciencedirect.com/topics/computer-science/inverse-document-frequency>
- <https://github.com/LGDoor/Dump-of-Simple-English-Wiki>