Raport z projektu na przedmiot Przetwarzanie Języka Naturalnego

Wyszukiwarka semantyczna parków narodowych w USA

Przemysław Kożuch
Radosław Brzostek
Jarosław Kołodziej

Abstrakt

Raport przedstawia sposób implementacji i wykonane testy wyszukiwarki semantycznej parków narodowych w Stanach Zjednoczonych Ameryki. Wykorzystuje ona różne rodzaje tokenizacji oraz wektoryzacji tekstu (TF-IDF, bag of words oraz word2vec). Korpus stanowią strony wikipedii o pojedynczych parkach narodowych z USA , przetworzone do formy tekstowej tak aby nie zawierały one elementów języka HTML czy innych niepotrzebnych znaków takich jak adnotacje. Poddawane są też one tokenizacji i lematyzacji, a następnie uproszczeniu przez algorytm PCA. Zbadana została zależność poprawności wyszukiwarki od użytej formy wektoryzacji, tokenizacji oraz ilości tematów w przypadku użycia TF-IDF.

Wstęp

Wyszukiwarka semantyczna to narzędzie pozwalające na wyszukiwanie elementu zbioru dokumentów (na przykład szeroko rozumianych stron internetowych) na podstawie nie tylko słów kluczowych i indeksowaniu tego zbioru, ale także na podstawie **znaczenia treści dokumentów**.

Klasyczne wyszukiwarki internetowe są to “programy generujące bazy danych używając botów internetowych [...] automatycznie śledzących linki i zbierających informacje na podstawie struktury HTML”[1], a odpowiedzi generowane użytkownikowi są w oparciu na “różnych algorytmach do indeksowania informacji w sieci”[1].

Tymczasem modele semantyczne skupiają się na analizie znaczenia zarówno dokumentów jaki i zapytania użytkownika. Uprzednio przetworzone dokumenty są sprowadzane różnego rodzaju wektorów opisujących ich treść w mniej lub bardziej dokładny sposób, gdzie każde słowo (lub lemat) ma swoją własną współrzędną. Następnie wektory te są upraszczane do kilkunastu wymiarów reprezentujących tematy. W tym celu wykorzystywana jest analiza głównych składowych, której celem jest “wydobycie najważniejszych informacji z tabeli danych; skompresowanie rozmiaru zestawu danych, zachowując tylko tą ważną informację; uproszczenie zbioru danych oraz analiza struktury obserwacji i zmiennych”[2]. Dzięki temu możliwe jest porównywanie takich wektorów między sobą za pomocą odległości kosinusowej [3], które jest tym większe im tematy dwóch dokumentów reprezentowanych przez nie są podobniejsze do siebie. Podobieństwo to wynika ze sposobu w jaki PCA upraszcza wektory.

Nowe osie “stają się raczej kombinacją częstotliwości słów niż częstotliwością jednego słowa. Można o nich myśleć jako o wadze słów w tworzeniu różnych tematów używanych w korpusie.” [4]

Model ten wymaga odpowiednio przetworzonego korpusu. Ponieważ słowa posiadające te samo znaczenie mogą przyjmować różne formy gramatyczne konieczna jest lematyzacja, a także sprowadzenie wszystkich liter do jednej wielkości np. do małych liter. Należy usunąć znaki interpunkcyjne. Również konieczne jest usunięcie tak zwanych “stop words”. Są to słowa pojawiające się bardzo często w danym języku, ale równocześnie nie przenoszące prawie żadnej informacji. [5]

Implementacja

Wyszukiwarka została zrealizowana przy użyciu języka python oraz bibliotek numpy, re, pandas, sklearn, nltk, gensim i matplotlib. Program został przedstawiony w postaci dokumentu jupyter notebook i składa się z kilku etapów zaimplementowanych w osobnych funkcjach:

- przetworzenie korpusu
    - tokenizacja
    - lematyzacja
- wektoryzacja
- trenowanie modelu
- przetworzenie zapytania użytkownika i wyświetlenie wyników

Przetworzenie korpusu

Na początku uruchamiany jest scrapper, który pobiera i przetwarza strony html o parkach narodowych w USA w języku angielskim i przetwarza je do formy tekstowej. Zapisuje je w tabeli wraz z linkiem do strony oraz tytułem artykułu (którym zawsze jest nazwa danego parku). Te informacje zostaną później wykorzystane w celu wyświetlenia wyników wyszukiwania. Ta tabela następnie jest zapisywana do pliku csv. Jeśli taki plik już istnieje w miejscu uruchomienia notebook’a, ten krok jest pomijany.

Przetworzone artykuły jednak nie są jeszcze gotowe gdyż zawierają one dużo informacji, które są niepotrzebne z perspektywy użytkownika i modelu wyszukiwania. Są one zawarte pomiędzy kwadratowymi nawiasami. Najczęściej są to odniesienia do źródeł wylistowanych na końcu każdego artykułu, które jednak nie są częścią naszego korpusu, dlatego muszą zostać usunięte. W takiej formie mogą również zostać przedstawione informacje dla redaktorów wikipedii takie jak to, że dana sekcja wymaga źródeł lub aktualizacji. W kwadratowych nawiasach również często znajduje się wymowa nazw własnych, która również nie jest potrzebna naszemu modelowi gdyż nie przewidujemy, że użytkownik nie będzie wprowadzał sposobu wymowy słowa używając symboli IPA.

Przykład fragmentu artykułu przed i po wyczyszczeniu:

[note 1][10][11] As of 2015[update], the permanent park boundary, as established by act of Congress in 1986, included 12,416 acres (19.4 sq mi; 50.2 km2) of privately owned land under conservation easements managed by the National Park Service, which plans to acquire the land at some point.
As of 2015, the permanent park boundary, as established by act of Congress in 1986, included 12,416 acres (19.4 sq mi; 50.2 km2) of privately owned land under conservation easements managed by the National Park Service, which plans to acquire the land at some point.

Tokenizacja i lematyzacja

Korpus został podzielony na zdania, które następnie za pomocą TreebankWordTokenizer z biblioteki nltk zostały podzielone na słowa. Zignorowane zostały znaki interpunkcyjne i powszechne słowa z języka angielskiego. Następnie każdy token został skrócony do jego najbardziej podstawowej formy zwanej rdzeniem używając lematyzera PorterStemmer również z biblioteki nltk. Do tego sposobu dalsza część dokumentu będzie odnosić się jako **sposobu głównego**.

Niektóre z poniżej opisanych wektoryzatorów posiadają wbudowane tokenizery, które również zostały użyte, a następnie porównane z głównym sposobem, co opisuje rozdział badania.

Wektoryzacja

W programie zostały zrealizowane trzy rodzaje wektoryzacji:

- TF-IDF
- bag of words
- word2vec

Aby istniała możliwość możliwość utworzenia oraz szybkiej zmiany sposobów wektoryzacji bez zmiany reszty kodu, różne rodzaje wektoryzacji zostały sprowadzone do wspólnego interfejsu. Wyjątek stanowi word2vecz biblioteki gensim posiadający własny moduł PCA i samodzielnie przetwarzająca query. Każda funkcja wektoryzująca przyjmuje nieprzetworzony korpus i przeprowadza prostą tokenizację lub przyjmuje gotowy korpus już przetworzony bardziej zaawansowaną formą tokenizacji. Zwracają one wektor powstały z korpusu oraz odniesienie do funkcji do przetwarzania zapytania w ten sam sposób. Dzięki temu zawołanie takiej funkcji pozwala na przetworzenie całego korpusu i równocześnie zastąpienie funkcji do wektoryzacji zapytania tak aby zwracała ona odpowiedni wektor:

```
# Vectorize:
data_vector, query_vectorizer = vectorizer(raw_corpus=corpus,\
tokenized_corpus=stem_corpus)
```

Do wektoryzacji TF-IDF został użyty pakiet sklearn, który umożliwia prostą tokenizację, która nie obejmuje upraszczania słów rdzenia, ale pozwala też na wgranie własnego korpusu

tokenów. Obie wersje zostały użyte i następnie porównane ze sobą w testach w sekcji badania.

Badania

W celu porównania skuteczności różnych rodzajów przetwarzania danych został utworzony prosty test badający ile z wprowadzonych wcześniej zapytań uzyskało poprawną odpowiedź. Zapytania te zostały utworzone ręcznie i posiadają jednoznaczną odpowiedź, np:

```
mountain goat is symbol of this park
```

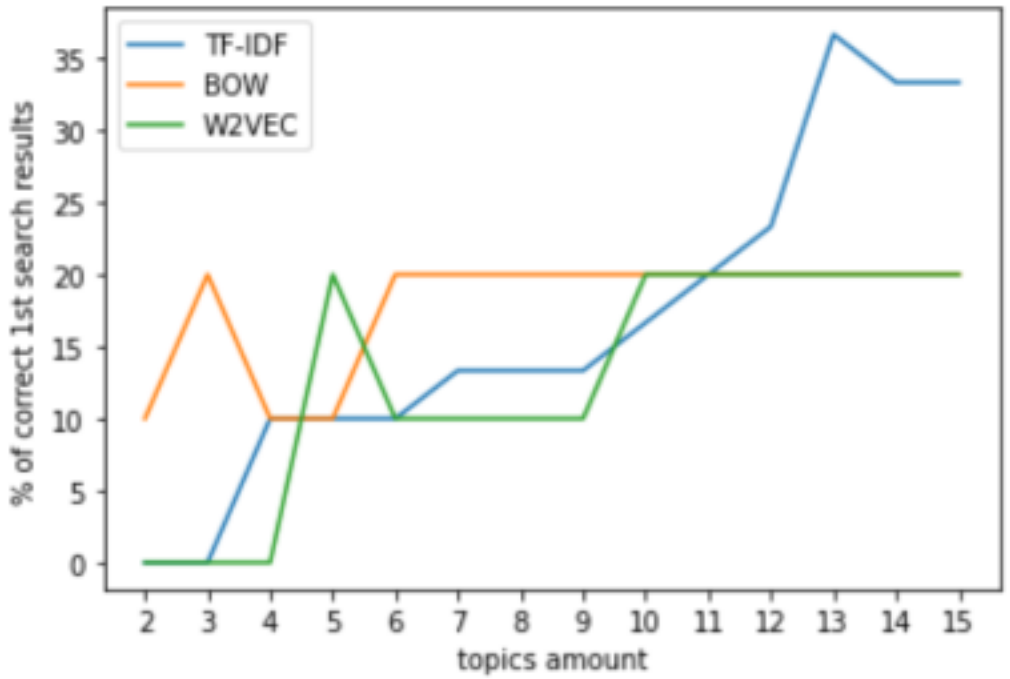
Aby przejść test najwyższą odpowiedzią do tego pytania powinno być Glacier National Park.

Lista wszystkich zapytań oraz odpowiedzi w postaci tytułów artykułów wikipedii znajduje się w pliku test_querries.txt

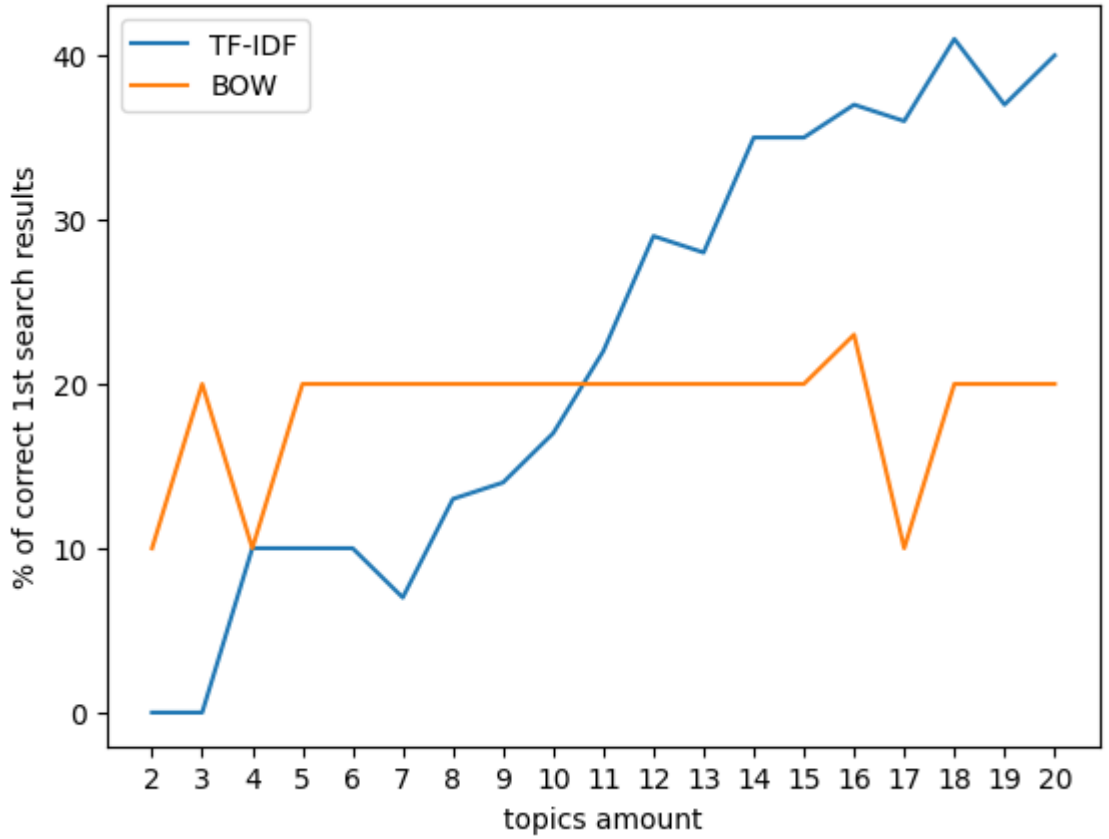
Test 1

W pierwszym teście porównane zostały ze sobą 3 sposoby wektoryzacji dla tak samo przetworzonego korpusu, po przeprowadzeniu tokenizacji i lematyzacji głównym sposobem. Również porównana została skuteczność algorytmu PCA dla różnych rozmiarów wektora wynikowego (ilości tematów).

Wynik testu:



Wyniki z większą liczbą tematów dla wektorów TF-IDF i BOW:



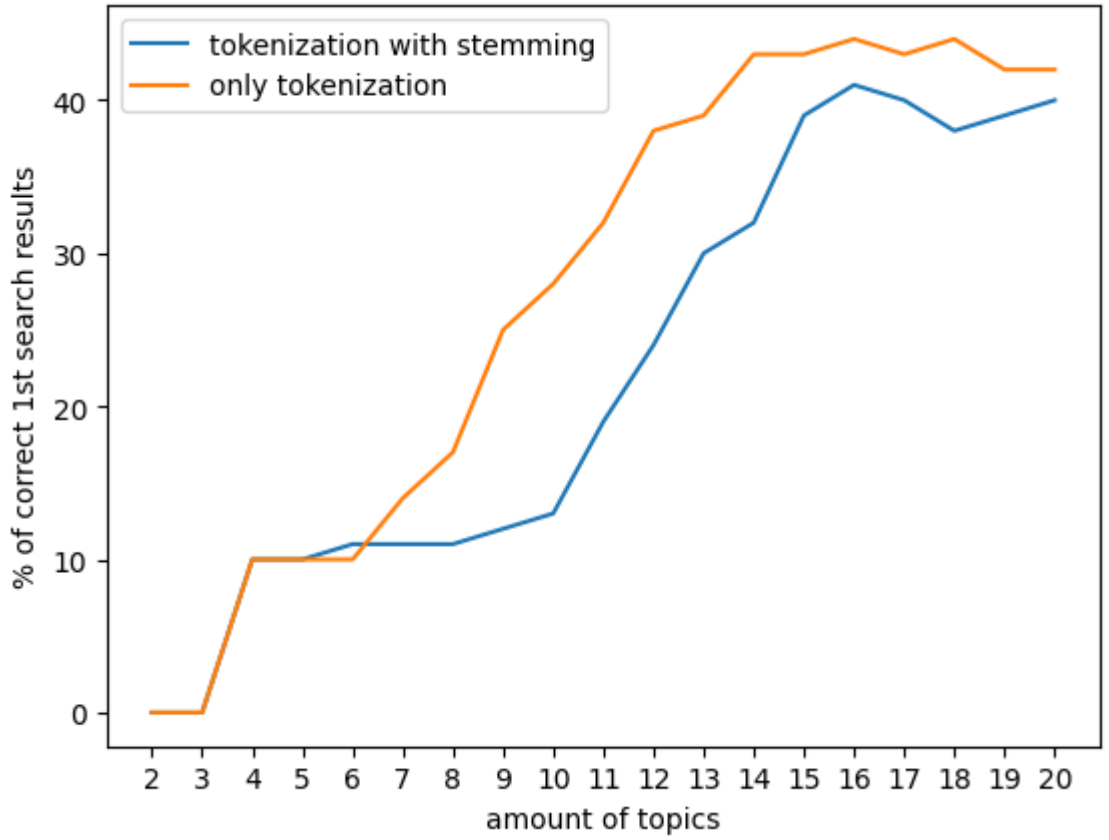
Wnioski:
Model z wektoryzacją TF-IDF zaczyna sobie radzić tym lepiej im więcej jest tematów, ale trend ten zwalnia w okolicach 16 tematów. Wektor BOW zdaje się nie ulepszać skuteczności wraz z liczbą tematów i stoi na poziomie około 20%.

Wektoryzacja word2vec w naszych testach wykazała trochę gorszą skutecznośc względem TF-IDF, ale widać, że rośnie wraz z ilością tematów.

Test 2

Ostatni test skupił się na porównaniu skuteczności wyszukiwarki w zależności od rodzaju przetworzenia korpusu. Przeciwko głównemu sposobowi został porównany domyślny tokenizer z wektoryzera TF-IDF z biblioteki nltk, który nie obejmuje lematyzacji. Liczba tematów wynosiła stałe x.

Wynik testu:



Wniosek:
Domyślna tokenizacja poradziła sobie ogólnie lepiej niż zaawansowana tokenizacja. Wynika z tego, że modele semantyczne nie wymagają lematyzacji, a wręcz może ona pogorszyć efektywność wyszukiwania, gdyż niektóre końcówki i informacje o gramatyce mogą być przydatne w rozumieniu znaczenia zdań.

Podsumowanie:

Z użytkowania poza testami można zauważyć, że wektoryzacja TF-IDF lepiej radzi sobie z krótkimi frazami np. “alaska” niż BOW, które nie potrafi przekazać znaczenia tych krótkich fraz i zaczyna pokazywać wyniki dopiero po wpisaniu dłuższych fraz zawierających wiele słów kluczy lub dosłownych fragmentów artykułów. Metoda używająca BOW miała problemy z wydobyciem najważniejszych części artykułu, które metoda z TF-IDF rozpoznaje nawet dla małych ilości tematów.

Ważne też jest dobre przygotowanie danych do wektoryzacji, wyczyszczenie ich z niepotrzebnych informacji oraz dobra tokenizacja. Domyślna tokenizacja okazała się więcej niż wystarczająca do naszych celów.

Bibliografia:

[1] C. Oppenheim, A. Morris , C. McKnight - (2000),"The evaluation of WWW search engines", Journal of Documentation
[2] Abdi, H., & Williams, L. J. (2010). *Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics,*
[3] https://en.wikipedia.org/wiki/Cosine_similarity - Cosine Similarity
[4] H. Lane, H. Hapke, C. Howard - Natural Language Processing in Action_ Understanding, analyzing, and generating text with Python (2019, Manning Publications)
[5] U. Krzeszewska , A. Poniszewska-Marańda, J. Ochelska-Mierzejewska - Systematic Comparison of Vectorization Methods in Classification Context,