

Wyszukiwarka semantyczna artykułów z Wikipedii

Aleksander T.

4 czerwca 2024

Spis treści

1	Wstęp	3
1.1	Cel	3
1.2	Zakres	3
1.3	Metodyka	3
2	Podstawy teoretyczne	3
2.1	TF-IDF	3
2.2	PCA	3
3	Implementacja projektu	4
3.1	Interfejs wyszukiwarki	4
3.2	Baza danych	7
3.3	API	8
3.4	Skrypty	9
3.5	Przetwarzanie tekstu na wektory TF-IDF i PCA	10
4	Podsumowanie	10

1 Wstęp

1.1 Cel

Celem projektu jest stworzenie wyszukiwarki semantycznej artykułów z anglojęzycznej Wikipedii o konkretnej tematyce. Wybrane zostało około 3000 artykułów z listy "Unusual Articles". Lista zawiera artykuły zaklasyfikowane przez wikipedystów jako nietypowe, ale jednocześnie wartościowe i zweryfikowane.

1.2 Zakres

Zakres projektu obejmuje stworzenie narzędzi do pobrania i przetworzenia artykułów, przygotowania bazy danych do przechowywania ich zwektoryzowanych form oraz zbudowanie API i interfejsu użytkownika, który umożliwi użytkownikowi przeszukiwanie artykułów w przystępny sposób.

1.3 Metodyka

W projekcie została wykorzystana metoda wektoryzacji dokumentów w oparciu o wektory TF-IDF połączona z redukcją wymiarów PCA. Do przetwarzania dokumentów został wykorzystany język Python i głównie biblioteka `sickit-learn`. Jako alternatywa dodane zostało również wyszukiwanie oparte o wektory wygenerowane przez rozwiązanie dostarczone przez firmę OpenAI.

2 Podstawy teoretyczne

2.1 TF-IDF

Pierwszy etapem przetwarzania tekstu w projekcie jest przetworzenie go na wektory TF-IDF[1][2]. W uproszczeniu wektory TF-IDF reprezentują istotność słowa dla danego dokumentu. W wyznaczaniu opisywanych wektorów brana jest pod uwagę częstotliwość występowania danego słowa we wszystkich badanych dokumentach. W rezultacie bardzo powszechne słowa będą miały z reguły małą istotność w większości artykułów, a słowa występujące tylko w jednym artykule będą dla niego najistotniejsze. Metoda ta może pozwolić na identyfikowanie tekstów które traktują o pewnych pojęciach w ponadprzeciętnym stopniu.

2.2 PCA

Następnym krokiem jest przetworzenie wektorów TF-IDF na wektory tematów, wykorzystując metodę PCA[3][2]. PCA pozwala na wyabstrahowanie ogólnych tematów charakteryzujących dane teksty. Wektor tematów to wektor informujący o tym, jak bardzo dany tekst jest związany z danym tematem. Na przykład tekst o kotach jest bardzo związany z hipotetycznym tematem "zwierzęcość", ale z tematem "elektroniczność" nie jest związany prawie w ogóle. Wektory utworzone przy pomocy metody PCA obrazują tę zależność. W przeciwieństwie

do przywołanego przykładu, w praktyce nie jest wiadome czego dokładnie dotyczy dany temat wyznaczony przez algorytm, ale taki przykład dobrze obrazuje opisywany mechanizm.

3 Implementacja projektu

Projekt składa się z czterech części: interfejsu wyszukiwarki w postaci aplikacji internetowej, bazy danych, API oraz szeregu skryptów służących do pobierania i przetwarzania danych. Wyszukiwanie odbywa się na podstawie wektorów TF-IDF, których wymiarowość została zredukowana przy wykorzystaniu metody PCA.

3.1 Interfejs wyszukiwarki

Interfejs wyszukiwarki został zaimplementowany w postaci aplikacji internetowej opartej o framework NextJS[4]. Poza główną opcją wyszukiwania artykułów oferuje także możliwość losowania artykułu i znajdowania artykułów podobnych do wybranego.



Rysunek 1: Główny widok wyszukiwarki

Użytkownik może wybrać metodę wyszukiwania artykułów. Główna i najszybsza metoda jest oparta o wspomniane wektory TF-IDF. Druga metoda dostępna na stronie, jest oparta o wektoryzację artykułów przy użyciu rozwiązań dostarczonych przez firmę OpenAI. Celem dodania drugiej metody było porównanie efektywności rozwiązania komercyjnego z rozwiązaniem opracowanym przy użyciu publicznie dostępnych bibliotek.

Gdy użytkownik wpisze w pole wyszukiwania tekst, dotyczący poszukiwanego tematu, aplikacja zwróci kilka artykułów najbardziej pasujących do zapytania. Artykuły sortowane są malejąco według podobieństwa do zapytania. Nawet jeśli znalezione zostaną artykuły bardzo odległe od zapytania, to i tak zostaną one zwrócone, ponieważ idea aplikacji zakłada, że użytkownik nie wie dokładnie, czego szuka, a nawet mało związane artykuły mogą być dla niego ciekawe.

UNUSUAL SEARCH

"Of the over six million articles in the English Wikipedia there are some articles that Wikipedians have identified as being somewhat unusual."

political parties

TFIDF

Search

Random Article

Full list

Results for "political parties"

Polish Beer-Lovers' Party 0.58

https://en.wikipedia.org/wiki/Polish_Beer-Lovers%27_Party

Political party in Poland

Donald Duck Party 0.65

https://en.wikipedia.org/wiki/Donald_Duck_Party

Political party in Sweden

Anarchist Pogo Party of Germany 0.72

https://en.wikipedia.org/wiki/Anarchist_Pogo_Party_of_Germany

Political party in Germany

Anti-PowerPoint Party 0.76

https://en.wikipedia.org/wiki/Anti-PowerPoint_Party

Swiss political party

Rysunek 2: Wyniki wyszukiwania

W wynikach wyszukiwania wyświetlany jest poziom podobieństwa artykułu do zapytania. Kliknięcie przycisku oznaczonego ikoną tyldy wyświetli podobne artykuły. Podobieństwo artykułów sprawdzane jest na podstawie wektorów TF-IDF i PCA.

5

Articles similar to Hungarian Two-Tailed Dog Party

Seijika Joshi 48 Party 0.87

https://en.wikipedia.org/wiki/Seijika_Joshi_48_Party

Political party in Japan

McGillicuddy Serious Party 0.87

https://en.wikipedia.org/wiki/McGillicuddy_Serious_Party

1984–1999 New Zealand satirical political party

Donald Duck Party 0.92

https://en.wikipedia.org/wiki/Donald_Duck_Party

Political party in Sweden

Nagriamel 0.98

<https://en.wikipedia.org/wiki/Nagriamel>

Political party in Vanuatu

Anti-PowerPoint Party 0.99

https://en.wikipedia.org/wiki/Anti-PowerPoint_Party

Swiss political party

Go back

Rysunek 3: Widok podobnych artykułów

Istnieje również opcja wylosowania artykułu. W tym celu należy kliknąć przycisk "Random Article", który znajduje się na stronie głównej aplikacji.

Your random article

Crime in Antarctica
https://en.wikipedia.org/wiki/Crime_in_Antarctica
Illegal acts in the world's southernmost continent

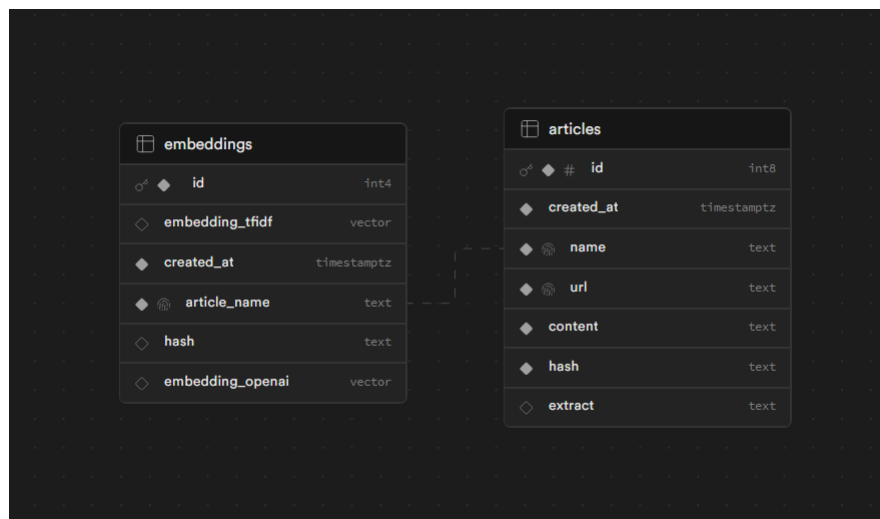
RerollGo back

Rysunek 4: Losowanie artykułu

Funkcja losowania artykułu nie jest oparta w żadnym stopniu o wektory TF-IDF.

3.2 Baza danych

Poniżej przedstawiona została struktura bazy danych.



Rysunek 5: Struktura bazy danych

W bazie danych przechowywane są treści wszystkich artykułów. W osobnej tabeli przechowywane są odpowiadające im wektory. Projekt korzysta z bazy danych PostgreSQL, a do przechowywania wektorów używany jest typ danych "vector", dostępny w rozszerzeniu "pgvector"[5]. Cały mechanizm wyszukiwania podobnych artykułów korzysta z oferowanego przez rozszerzenie operatora

obliczającego cosinusowy dystans między wektorami. Cosinusowy dystans jest miarą podobieństwa między dwoma wektorami.

3.3 API

API zostało zaimplementowane w języku Python przy użyciu frameworka Flask[6]. API wczytuje modele, zapisane w formacie pickle i wykorzystuje je do wektoryzacji zapytań użytkownika.

```
1 def tfidf_search(query):
2     vec = vectorizer.transform([query]).toarray()
3     if not vec[0].any():
4         return {
5             "message": "No matching articles found"
6         }, 404
7
8     vec = pca.transform(vec)[0]
9     with psycopg.connect(config["PG_URL"]) as conn:
10         plain_list = vec.tolist()
11         results = []
12         with conn.cursor() as cur:
13             cur.execute("""SELECT article_name,
14                             round((embedding_tfidf <=> %s::vector)::
15 numeric, 2) AS distance,
16                             url, extract
17                             FROM unusual_search.embeddings
18                             left join unusual_search.articles on
19                             article_name = name
20                             ORDER BY distance ASC LIMIT 10""", (
21 plain_list,))
22             embeddings = cur.fetchall()
23
24             for embedding in embeddings:
25                 results.append({
26                     "article_name": embedding[0],
27                     "distance": embedding[1],
28                     "url": embedding[2],
29                     "extract": embedding[3]
30                 })
31
32     return results
```

Listing 1: Funkcja znajdująca artykuł podobny do zapytania

W przypadku zapytania o podobne artykuły, API wybiera z bazy danych artykuły najbardziej podobne do wskazanego artykułu. W tym przypadku wektory TF-IDF nie są obliczane na bieżąco: używane są wektory zapisane w bazie.

```
1 @app.route("/similar")
2 def similar_articles():
3     article_name = request.args.get('article_name')
4     if not article_name:
5         return {
6             "message": "Missing article_name"
7         }, 400
8
9     with psycopg2.connect(config["PG_URL"]) as conn:
10        with conn.cursor() as cur:
11            cur.execute("""select article_name,
12                               round((embedding_tfidf <=> (select
13                               embedding_tfidf from unusual_search.embeddings where
14                               article_name = %s))::numeric, 2) as distance,
15                               url, extract
16                               from unusual_search.embeddings
17                               left join unusual_search.articles on
18                               article_name = name
19                               order by distance asc
20                               limit 5
21                               offset 1
22                               """, (article_name,))
23
24            results = []
25
26            for article in cur.fetchall():
27                results.append({
28                    "article_name": article[0],
29                    "distance": article[1],
30                    "url": article[2],
31                    "extract": article[3]
```

Listing 2: Funkcja znajdująca artykuły podobne do innego artykułu

3.4 Skrypty

Skrypty posłużyły do wygenerowania wektorów TF-IDF oraz przetworzenia ich przy użyciu PCA, a następnie zapisania ich w bazie danych. Skrypty zostały napisane w języku Python i korzystają z bibliotek takich jak scikit-learn, psycopg2, pandas, numpy. Działanie skryptów opisane jest w sekcji 3.5.

3.5 Przetwarzanie tekstu na wektory TF-IDF i PCA

Do stworzenia wektorów TF-IDF została wykorzystana biblioteka scikit-learn, a w szczególności klasa `TfidfVectorizer`, która obsługuje również krok tokenizacji i usuwania słów stopowych, w tym przypadku dla języka angielskiego. Stemming i lematyzacja nie zostały zastosowane. Słowa, które występują w mniej niż 10 artykułach, zostały pominięte, ponieważ założone jest, że użytkownik będzie szukał artykułów używając ogólnych pojęć i zdań, a nie konkretnych słów kluczowych.

```
1 vectorizer = TfidfVectorizer(stop_words='english', min_df=10)
2 vectorizer.fit(texts)
3 vecs = vectorizer.transform(texts)
```

Natomiast poniżej przedstawione zostały wynikowe wektory TF-IDF.

	00	000	001	002	004	007	01	...	zu	zur	zürich	émile	état	über	ha
52-hertz_whale	0.00000	0.00000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Bahia_incident	0.02132	0.00000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Arm-Fall-Off-Boy	0.00000	0.00589	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Arrhichion	0.00000	0.00000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Boston_Typewriter_Orchestra	0.00000	0.00000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Tablica "texts" zawiera treści artykułów. Wynikowe wektory "vecs" są następnie przekazywane do obróbki przy użyciu PCA.

```
1 pca = PCA(n_components=0.95)
2 pca.fit(vecs.toarray())
3 pca_vecs = pca.transform(vecs.toarray())
```

Warto zwrócić uwagę, że liczba komponentów PCA (wynikowa długość wektora) została wybrana tak, by wyjaśniać co najmniej 95% wariancji danych[7]. Poniżej został przedstawiony efekt przejścia danych przez PCA.

	PCA0	PCA1	PCA2	PCA3	...	PCA2605	PCA2606	PCA2607	PCA2608
52-hertz_whale	0.011716	0.000625	-0.076410	0.085735	...	0.004533	0.004944	0.011173	0.007013
Bahia_incident	-0.047358	0.035676	0.078935	-0.053133	...	0.002675	0.015293	0.011155	0.013912
Arm-Fall-Off-Boy	0.027157	0.071954	-0.037769	-0.091348	...	0.004877	0.006698	0.011752	-0.007250
Arrhichion	0.002715	-0.034423	-0.016036	-0.029755	...	0.007465	-0.001644	0.015150	-0.003346
Boston_Typewriter_Orchestra	0.021261	0.137421	-0.045881	0.040956	...	0.002565	0.008635	-0.006409	-0.007475

Dużą zaletą PCA, poza dodawaniem warstwy semantycznej do wyszukiwania, jest znaczne ograniczenie wymiaru wektora, co w tym przypadku pozwoliło na przejście z długości 16878 do 2609.

Proces został przeprowadzony dla wszystkich (około 3000) artykułów, a wynikowe wektory zostały zapisane w bazie danych. Analogiczny proces odbywa się w API, gdy zostanie do niego podane zapytanie użytkownika, ale w tym przypadku wykorzystywane są zapisane modele PCA i TF-IDF, które ładowane są z plików pickle przy starcie aplikacji.

4 Podsumowanie

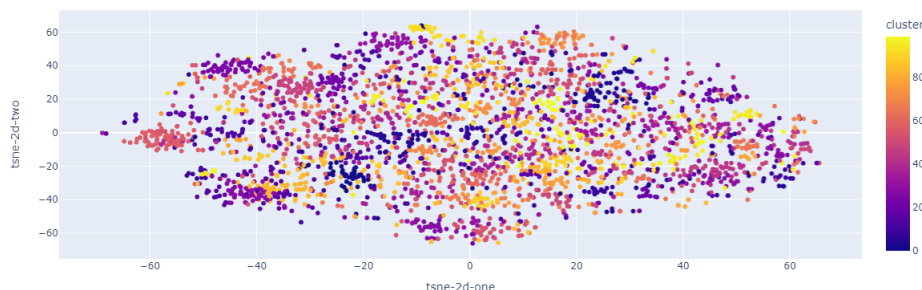
Projekt został zrealizowany w pełni w zakresie swoich podstawowych założeń (wyszukiwanie semantyczne metodą wektorów TF-IDF i PCA), a apli-

kacja internetowa została wdrożona i jest publicznie dostępna pod adresem <https://unusualsearch.org/>.

Możliwe kierunki rozwoju projektu to:

- Automatyzacja procesu aktualizacji artykułów/dodawania nowych,
- Dodanie innych metod wyszukiwania,
- Optymalizacja modeli PCA i TF-IDF, a w szczególności optymalizacja parametru `min_df`, który w obecnej formie może pomijać niektóre istotne, choć rzadziej występujące słowa.

W ramach projektu artykuły zostały również zwizualizowane w postaci wykresu.



Kolory na wykresie to przyporządkowane grupy tematyczne, a współrzędne X i Y zostały otrzymane poprzez redukcję wymiarów wektorów poprzez metodę t-SNE.

Bibliografia

- [1] *TF-IDF* - *Wikipedia*. URL: <https://en.wikipedia.org/wiki/Tf%E2%80%93idf> (visited on 06/01/2024).
- [2] H. Lane, H. Hapke, and C. Howard. *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python*. Manning Publications, 2019. ISBN: 9781617294631. URL: <https://books.google.pl/books?id=UyHgswEACAAJ>.
- [3] *Principal component analysis* - *Wikipedia*. URL: https://en.wikipedia.org/wiki/Principal_component_analysis (visited on 06/01/2024).
- [4] *NextJS* - *Documentation*. URL: <https://nextjs.org/docs> (visited on 06/01/2024).
- [5] *pgvector* - *Open-source vector similarity search for Postgres*. URL: <https://github.com/pgvector/pgvector> (visited on 06/01/2024).

- [6] *Flask - Documentation*. URL: <https://flask.palletsprojects.com/en/3.0.x/> (visited on 06/01/2024).
- [7] Bartosz Mikulski. *PCA - how to choose the number of components?* 2019. URL: <https://mikulskibartosz.name/pca-how-to-choose-the-number-of-components> (visited on 06/01/2024).