19.12.2021, Kraków

Wyszukiwarka semantyczna z bazą opartą na Wikipedii

Opracowali:
Anna Baran
Karol Baran
Michał Mosoń
Paweł Stachula
Rafał Wałkowski

Cel projektu

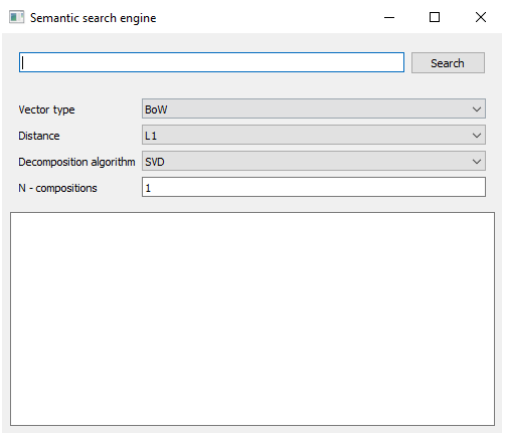
Celem tego projektu było opracowanie wyszukiwarki semantycznej, która pozwoli na przeszukiwanie Wikipedii pod kątem jednego, wybranego przez twórców projektu schematu. Projekt koncentruje się na wykorzystaniu i istocie semantyki w efektywnym przetwarzaniu i interpretowaniu informacji wprowadzanych przez użytkownika. W efekcie powinniśmy uzyskać aplikację, będącą w stanie zinterpretować w poprawny sposób zapytanie użytkownika i wyświetlić mu poprawne podpowiedzi.

Zakres pracy

W zakres projektu wchodziło określenie wymagań potrzebnych do zrealizowania projektu. Następnie należało wybrać zakres tematyczny, w którym operować będzie wyszukiwarka semantyczna, a także złożoność algorytmów wykorzystanych przy tworzeniu aplikacji. Kolejnym krokiem była implementacja rozwiązania w języku python. Zdecydowano się na opracowanie formy graficznej, będącej rozwiązaniem bardziej przyjaznym dla użytkownika. W takim wypadku potrzebne było zarówno zaprojektowanie warstwy graficznej, jak i logiki, na której opierało się działanie aplikacji.

Opis rozwiązania

Podstawą utworzenia wyszukiwarki semantycznej było dobranie odpowiedniego zaplecza warstwy graficznej. W tym projekcie zdecydowano się na bibliotekę PyQt, która pomaga oddzielić warstwę widoku od logiki. Jedną z wad tego rozwiązania było to, że aplikacja jest darmowa tak długo, jak długo jest oferowana na licencji open-source. Ze względu na charakterystykę projektu, takie wymaganie nie było ograniczeniem, dlatego zdecydowano się na wykorzystanie bardziej zaawansowanego i przyjaznego programiście oprogramowania. W efekcie aplikacja prezentuje się w sposób następujący:



Możliwe są 4 modyfikatory wpływające na wyniki wyszukiwania. Pierwszym z nich jest wybór typu wektora. Są to dwa rodzaje, gdzie pierwszym jest Bag Of Words, używany zazwyczaj jako narzędzie pozwalające na generowanie cech. Dzięki temu możliwe jest charakteryzowanie tekstu, dalej wykorzystywane jako pomoc w rozpoznawaniu

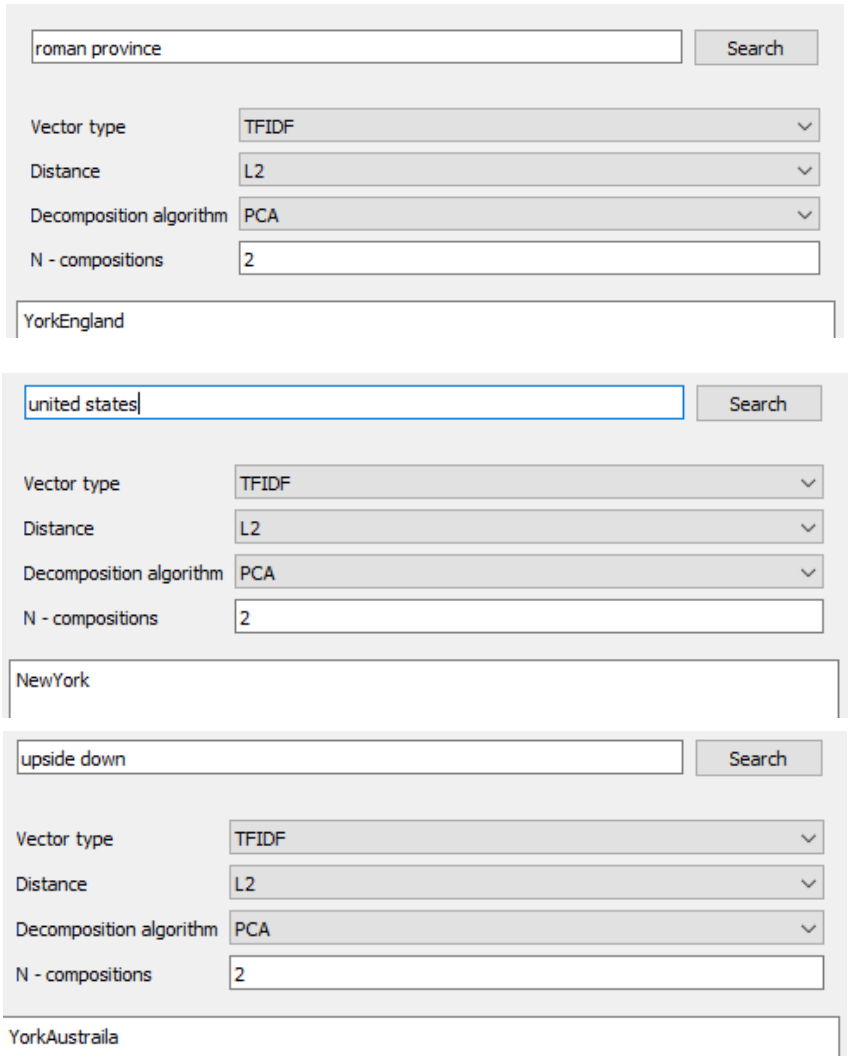
odpowiednich słów kluczowych. Następnie użytkownik wybiera rodzaje dystansu, najlepsze rezultaty powinny dawać opcje L2 i cosine, gdzie L2 oznacza odległość Euklidesowską. Pomimo uzyskiwania najlepszych wyników, niewykluczone są rozbieżności pomiędzy tymi dwoma wyborami. Do dekompozycji wybrano trzy algorytmy - PCA, SVD i LDiA. Podczas gdy PCA poszukuje atrybutów o najwyższej wariancji, celem LDA jest zmaksymalizowanie podziału pomiędzy kategoriami, a SVD rozkłada macierze na iloczyn trzech specyficznych macierzy, redukując tym samym ich wymiar. Na koniec użytkownik wybiera liczbę kompozycji, gdzie wybrana liczba powinna znajdować się w zakresie od 1 do N.

```
def __init__(self, distance, vector_kind, decomposition_algorithm, number_of_components):
    self.distance = lambda x, y: pairwise_distances(x, y, distance)
    if vector_kind == 'bow':
        self.vectorizer = CountVectorizer()
    elif vector_kind == 'tfidf':
        self.vectorizer = TfidfVectorizer()

    if decomposition_algorithm == 'svd':
        self.da = TruncatedSVD(n_components=number_of_components)
    elif decomposition_algorithm == 'pca':
        self.da = PCA(n_components=number_of_components)
    elif decomposition_algorithm == 'ldia':
        self.da = LDiA(n_components=number_of_components)

    self.corpus = []
    self.labels = []
    self.corpusVect = []
    self.topics = []
    self.question = ''
```

Udane zaprojektowanie aplikacji opierało się na sprawnym zaimplementowaniu już dostępnych rozwiązań, oferowanych przez bibliotekę sklearn. Wyzwanie stanowiło połączenie ogólnodostępnych rozwiązań w jeden działający system wykorzystujący swoją wiedzę do proponowania użytkownikowi nie tylko konkretnie wyszukiwanych fraz, ale też zrozumienia ogólnego kontekstu i intencji osoby szukającej.

Testowanie i Wyniki

Aby skutecznie sprawdzać aplikację na etapie tworzenia algorytmów, były potrzebne dane pozwalające na łatwe i szybkie testy. Do tego celu wykorzystano miasta York, znajdujące się kolejno w Ameryce, Anglii i Australii. W celu poprawnej weryfikacji była wymagana dobra znajomość każdego z tekstów wejściowych. Gdy wszystkie wykorzystywane algorytmy działały poprawnie, nadeszła pora na ich optymalizację. Niektóre z parametrów lepiej radziły sobie przy bezpośrednim wyszukiwaniu, ale gubiły się przy mniej dosadnych wyszukiwaniach. Metodą prób i błędów ustalono, że najlepszym jest rozwiązanie bazujące na wektorze TFIDF, z dystansem Euklidesowskim, wykorzystujące algorytm PCA i dwie kompozycje.

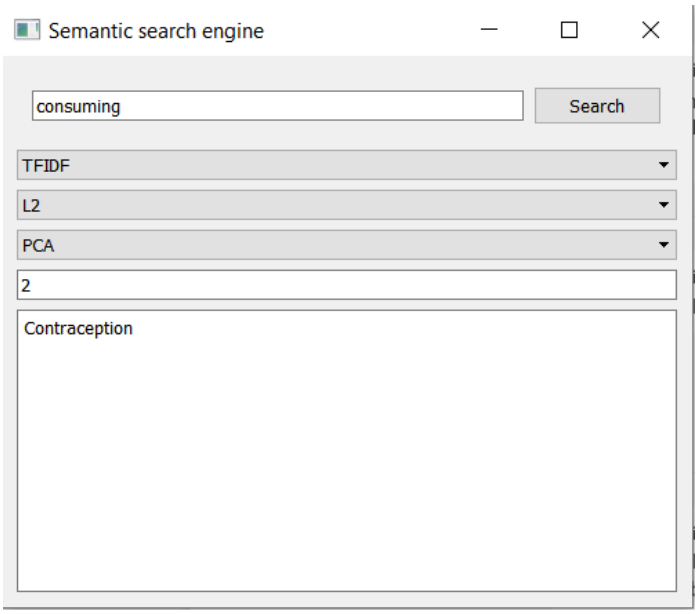


Ta optymalizacja poradziła sobie nawet z odgadnięciem, które z miast jest “do góry nogami”, przynosząc tym samym zaskakująco dobre rezultaty. Wikipedia, stanowiąc ogromne skupisko wiedzy, okazała się być zbyt dużym przedsięwzięciem by możliwe było przetestowanie utworzonych algorytmów na całej bazie danych. Otrzymane rezultaty napawają jednak optymizmem, pokazując potencjał i wyższość wyszukiwania semantycznego ponad tradycyjnym hasłowym wyszukiwaniem. Dzięki temu projektowi udało nam się lepiej zrozumieć znaczenie semantyki i jej wykorzystanie we współczesnym świecie.

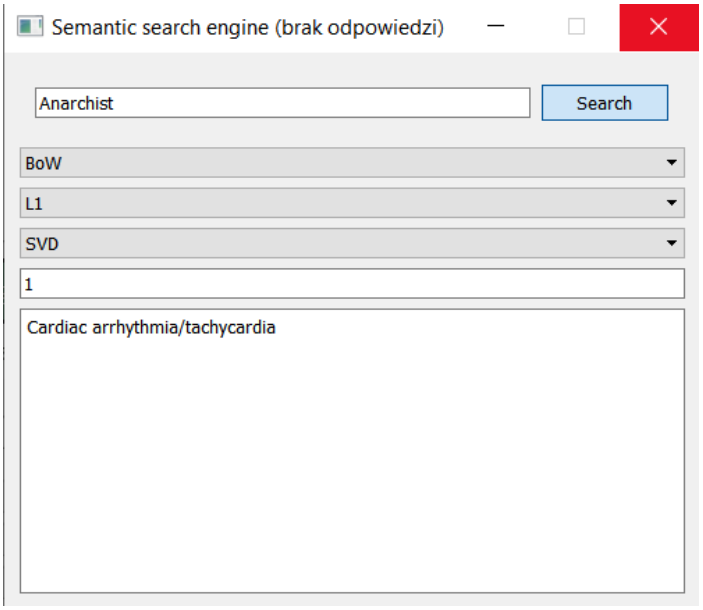
Następnym krokiem było przetestowanie wyszukiwarki na większej części danych z Wikipedii. W tym celu pobrano z https://dumps.wikimedia.org/backup-index.html plik w formacie XML zawierający dane z całej strony. Dane po rozpakowaniu miały wielkość 80,6GB. Z uwagi na ten rozmiar, załadowanie całości pliku nie było możliwe, zamiast tego wykorzystano funkcję *iterparse* z biblioteki ‘xml.etree.ElementTree’ aby iterować po tagach i uzupełniać tablice korpusu oraz tytułów. Zdecydowano się na wykorzystanie jedynie części definicyjnej artykułów aby zmniejszyć ilość danych dla każdego artykułu. Usunięto również wszystkie hasła przekierowujące do innej strony.

```python
for event, elem in iterparse(file_path, events=("start", "end")):
    if elem.tag == "{http://www.mediawiki.org/xml/export-0.10/}page" and event == "start":
        page_flag = True
    if page_flag:
        if elem.tag == "{http://www.mediawiki.org/xml/export-0.10/}title" and event == "end":
            title_good = True
            title.append(elem.text)
        if elem.tag == "{http://www.mediawiki.org/xml/export-0.10/}text" and event == "end":
            parsed = elem.text
            if parsed.startswith('#REDIRECT'):
                title.pop()

            else:
                parsed = re.sub("\{\{.*?\}\}", "", parsed)
                parsed = re.sub("\n", "", parsed)
                parsed = parsed.split("==")[0]
                parsed = re.sub("\\'", "", parsed)
                parsed = re.sub("(\[\[Category\:.*?\]\])|(\[\[File\:.*?\]\])", "", parsed)
                parsed = re.sub("\[\[", "", parsed)
                parsed = re.sub("\|.*?\]\]", "", parsed)
                parsed = re.sub("\]\]", "", parsed)
                parsed = re.sub("\<\!\-\-.*?\-\-\>", "", parsed)
                parsed = re.sub("\<ref\>.*?\<\/ref\>", "", parsed)
                # parsed = re.sub("\=\= See also \=\=.*", "aa", parsed)
                corpus.append(parsed.lower())
```

Aby skrócić czas wyszukiwania nie zdecydowano się na załadowanie całości danych, a pierwszych 3850 artykułów (5000 przed odrzuceniem przekierowań). Przykładowy wynik wyszukiwania na takim korpusie wygląda następująco:



Przykład 1



Przykład 2

Jak widać wyniki w tym przypadku nie są tak zadowalające jak przy danych testowych. Jeśli chodzi o pierwszy przykład możemy doszukać się logicznego powiązania, tak w drugim przypadku będzie to dużo trudniejsze. Powodów takiej sytuacji może być kilka. Przede wszystkim artykuły nie są związane z jednym konkretnym tematem, jest to jedynie wycinek artykułów które mogą dotyczyć wszystkiego, dlatego nie do końca wiadomo jakie wyniki mogą być zwrócone i jakie hasła w ogóle mają sens wykorzystania. Jeśli wpiszemy hasło niepowiązane z żadnym wczytanym artykułem możemy otrzymać nie za dobry wynik z winy niepełnych danych.

Kolejnym problemem który został zidentyfikowany, i który odpowiada za wynik wyszukiwania na drugim przykładzie są “Disambiguation Page”, czyli strony zbierające wiele haseł niepowiązanych ze sobą logicznie, oprócz posiadania na przykład takiego samego skrótu.

AF

From Wikipedia, the free encyclopedia

AF, **af**, **Af**, etc. may refer to:

Arts and entertainment [edit]

- Af, one of the Angels of Punishment, a mythical character outlined in Gustav Davidson's 1967 book *A Dictionary of*
- A.F. (band), a Swiss punk rock band started in 1992 and originally named Allpot Futsch
- A-F Records, an independent record label in Pittsburgh, Pennsylvania, US, founded by the band Anti-Flag
- *Almost Family* episode titles tend to be "*[Adjective]* AF"

Businesses and organizations [edit]

European [edit]

- ÅF, a Swedish technical consulting company
- AF Gruppen, a multinational construction and development company based in Norway
- Académie française, the official institution responsible for overseeing the French language
- Action Française, a French far right political movement
- Anarchist Federation (British Isles), an Anarchist-Communist agitational organisation in Britain

International [edit]

- Abercrombie & Fitch, an American-based, international clothing retailer
- Air France (IATA airline code and Euronext stock symbol "AF")
- The Adaptation Fund, a UN organization responsible for climate change adaptation
- Adventist Forums, an organization of progressive Seventh-day Adventists
- Alliance Française, an international organization that aims to promote French language and culture

Elsewhere [edit]

- American Freightways, former American trucking company in 2006 merged into FedEx Freight

- Autofocus, the ability of a camera to focus automatically
- Anisotropic filtering

Other uses in science and technology [edit]

- Af, the Köppen climate classification for a tropical rainforest climate
- Across flats (A/F), a measure of hexagonal nut flat size
- Advanced Format, a new disk format and access using sector sizes larger than 512 bytes
- Anaerobic filter, a type of anaerobic digester
- Arcuate fasciculus, a nerve bundle in the brain
- Atrial fibrillation, a form of cardiac arrhythmia

"Disambiguation Page" dla wyszukiwania w Przykładzie 2

W wykorzystanym zrzucie bazy Wikipedii strony typu "Disambiguation" nie są w żaden sposób oznaczone, co może zaburzać wyniki wyszukiwania tak jak w podanym powyżej przykładzie.