Dokumentacja: Przetwarzanie jezyka naturalnego w systemach sztucznej inteligencji - Projekt I

Autor: Aleksandra Ignacyk

1 Przeglad

Celem tego projektu jest przetworzenie i analiza danych tekstowych pozyskanych z artykułów Wikipedii. W szczególności projekt obejmuje nastepujace zadania:

1. Pobieranie danych tekstowych z artykułów Wikipedii,
2. Przetwarzanie danych (czyszczenie i normalizacja tekstu),
3. Analiza rozkładu czestotliwości słów, w tym wizualizacja przy użyciu Prawa Zipfa,
4. Budowanie grafów zależności miedzy słowami i analiza najważniejszych wezłów.

W niniejszej dokumentacji omówiono poszczególne kroki wraz z wyjaśnieniem kluczowych funkcji i fragmentów kodu.

2 Importowanie bibliotek

Pierwszym krokiem jest zaimportowanie bibliotek, które sa niezbedne do wykonania wszystkich zadań zwiazanych z przetwarzaniem tekstu i analiza danych.

- `requests` - służy do pobierania danych z URL,
- `BeautifulSoup` (z `bs4`) - do parsowania kodu HTML i wyodrebniania treści z artykułów,
- `Counter` (z `collections`) - do zliczania czestotliwości wystepowania słów,
- `matplotlib.pyplot` i `numpy` - do wizualizacji danych,
- `re` - do czyszczenia tekstu (usuwanie liczb i znaków specjalnych),
- `json` - do przetwarzania odpowiedzi z API w formacie JSON,
- `networkx` - do budowania grafów i analizy zależności miedzy słowami.

Kod importujacy te biblioteki:

```
import requests
from bs4 import BeautifulSoup
from collections import Counter
import matplotlib.pyplot as plt
import numpy as np
import re
import json
import networkx as nx
```

3 Pobieranie treści z URL

Aby pobrać treść artykułu, implementujemy funkcje `get_page_content`, która:

- Wysyła żadanie HTTP do podanego URL,
- Wyodrebnia treści z wszystkich paragrafów `<p>` w kodzie HTML przy pomocy `BeautifulSoup`,
- Zwraca zebrany tekst lub pusty ciag w przypadku niepowodzenia żadania.

Kod funkcji:

```
def get_page_content(url):
    response = requests.get(url)
    if response.status_code == 200:
        soup = BeautifulSoup(response.content, 'html.parser')
        paragraphs = soup.find_all('p')
        text = ' '.join([p.get_text() for p in paragraphs])
        return text
    return ""
```

4 Przetwarzanie tekstu

Kolejnym krokiem jest przetworzenie pobranego tekstu, aby był gotowy do analizy. Funkcja `preprocess_text` wykonuje nastepujace operacje:

- Usuwa liczby za pomoca wyrażeń regularnych,
- Usuwa znaki specjalne, pozostawiajac jedynie litery i spacje,
- Zamienia tekst na małe litery i dzieli go na liste słów.

Kod funkcji:

```
def preprocess_text(text):
    text = re.sub(r'\d+', '', text) # Usuni cie liczb
    text = re.sub(r'[^a-zA-Z                    \s]', ' ',
       text) # Usuni cie znak w specjalnych
    return text.lower().split()
```

5 Pobieranie artykułów z API Wikipedii

Korzystajac z API Wikipedii, pobieramy do 100 artykułów. Proces obejmuje:

- Wybór jezyka wyszukiwania (w tym przypadku polski, czyli `pl`),
- Wysłanie żadania do API Wikipedii, przetwarzanie odpowiedzi w formacie JSON,
- Zapisanie linków do pełnych wersji artykułów w liście `urls`.

Kod:

```
language_code = 'pl'
search_query = 'the'
number_of_results = 100
headers = {
    'User-Agent': 'YOUR_APP_NAME (YOUR_EMAIL_OR_CONTACT_PAGE)'
}

base_url = 'https://api.wikimedia.org/core/v1/wikipedia/'
endpoint = '/search/page'
url = base_url + language_code + endpoint
```

```
parameters = {'q': search_query, 'limit': number_of_results}

response = requests.get(url, headers=headers, params=parameters)
response = json.loads(response.text)

urls = []
for page in response['pages']:
    article_url = 'https://' + language_code + '.wikipedia.org/wiki/' +
        page['key']
    urls.append(article_url)

print(urls)
```

6 Analiza czestotliwości słów

Na podstawie pierwszego artykułu pobranego z listy `urls`, przeprowadzamy analize rozkładu słów:

- Pobieramy treść artykułu i przetwarzamy tekst na liste słów,
- Zliczamy wystapienia słów przy użyciu `Counter`,
- Tworzymy wykres rozkładu czestotliwości słów zgodnie z Prawem Zipfa.

Kod:

```
url = urls[0]
text = get_page_content(url)

if text:
    words = preprocess_text(text)
    word_count = Counter(words)
    sorted_words = sorted(word_count.items(), key=lambda x: x[1], reverse=
        True)

# Prawo Zipfa - przygotowanie danych
ranks = np.arange(1, len(sorted_words) + 1)
freq = np.array([count for _, count in sorted_words])

# Wykres log-log (prawo Zipfa)
plt.loglog(ranks, freq, marker="o")
plt.title("Prawo Zipfa - zale no    rangi a cz stotliwo ci")
plt.xlabel("Ranga")
plt.ylabel("Cz stotliwo  ")
plt.grid(True)
plt.show()
```

7 Tworzenie grafu słów

Na końcu budujemy graf zależności miedzy słowami na podstawie ich sasiedztwa w tekście. Tworzymy graf, w którym krawedź wystepuje, jeśli dwa słowa sa sasiadujace w artykule.

- Dodajemy krawedzie miedzy sasiadujacymi słowami w każdym artykule,
- Wyodrebniamy 10% wezłów o najwiekszej liczbie sasiadów i wizualizujemy powstały graf.

Kod:

```
G = nx.Graph()
G.add_nodes_from(all_words)

# Dodanie kraw dzi mi dzy s owami wyst puj cymi obok siebie
```

```
for url in urls:
    text = get_page_content(url)
    if text:
        words = preprocess_text(text)
        for i in range(len(words) - 1):
            word1 = words[i]
            word2 = words[i + 1]
            G.add_edge(word1, word2)

# Wyodrbnienie 10% wz w o najwikszej liczbie ssiadw
num_nodes_to_extract = int(0.1 * len(G.nodes()))
sorted_nodes_by_degree = sorted(G.degree, key=lambda x: x[1], reverse=True)
top_nodes = [node for node, degree in sorted_nodes_by_degree[:
    num_nodes_to_extract]]

# Tworzenie podgrafu
H = G.subgraph(top_nodes)

# Wizualizacja grafu
plt.figure(figsize=(12, 12))
pos = nx.spring_layout(H, k=0.5)
nx.draw(H, pos, with_labels=True, node_size=1000, node_color="lightblue",
    font_size=8, font_weight="bold")
plt.title("Graf 10% wz wz najwikszej liczbsiadw (100
    artykuw)")
plt.show()
```

8 Wnioski

Projekt przedstawia sposób pobierania, przetwarzania i analizy tekstu z artykułów Wikipedii. Zastosowano Prawo Zipfa do zrozumienia rozkładu słów oraz narzedzia grafowe do wizualizacji zależności miedzy słowami. Możliwości analizy tekstu w naturalnych jezykach pozwalaja na odkrywanie interesujacych zależności, co ma szerokie zastosowanie w systemach sztucznej inteligencji.

9 Link do Githuba

https://github.com/olaignacyk/przetwarzanie$_{j}ezyka/blob/main/zadanie1 - 3.ipynb$