Narzędzie do znajdywania podobnych pytań na Stack Overflow

Mizak Mateusz, Mazur Szymon

1. Wstęp

Celem napisanego przez nas narzędzia jest odnajdywanie pytań, podobnych do pytania zadanego przez użytkownika, znajdujących się na Stack Overflow. Po podaniu przez użytkownika konkretnego pytania, program daje propozycje najbardziej podobnych do niego pytań już zadanych na Stack Overflow.

2. Rozwinięcie

Z powodów problemów związanych ze zbiorem danych udostępnionych na stronie www.kaggle.com postanowiliśmy sami stworzyć prosty skrypt, który umożliwiłby nam pobranie bazy pytań Stack Overflow za pomocą otwartego API stackexchange.

```python
import requests

api_url = "https://api.stackexchange.com/2.2/questions"

params = {
    "order": "desc",
    "sort": "votes",
    "site": "stackoverflow",
    "tagged": "python",
    "pagesize": 100
}
num_questions = 2500
pagesize = 100
num_pages = num_questions // pagesize + (num_questions % pagesize > 0)
questions = []
for page in range(1, num_pages + 1):
    params["page"] = page
    print(page)
    response = requests.get(api_url, params=params)
    if response.status_code == 200:
        page_questions = response.json()["items"]
        questions += page_questions
    else:
        print("Error retrieving questions:", response.status_code)
        break

with open("questions.txt", "a", encoding='utf-8') as f:
    i = 0
    for question in questions:
        f.write(str(i) + ". " + question["title"] + "\n")
        i += 1

```

Zbiór danych pobrany w taki sposób nie jest tak duży jak ten znajdujący się w źródle podanym przez prowadzącego. Jest to spowodowane ograniczeniami nałożonymi przez stackexchange na ilość zapytań, które API może otrzymać od pojedynczego użytkownika. Metodą na ominięcie tej blokady było by zarejestrowanie swojej aplikacji. Mimo stosunkowo małego zbioru danych jest on wystarczający, aby zaprezentować działanie naszego narzędzia.

Przykład jak wygląda wygenerowany przez nas zbiór danych:

```
1634. How to check if a float value is a whole number
1635. How do I print the model summary in PyTorch?
1636. Convert tuple to list and back
1637. Python type hinting without cyclic imports
1638. SQLAlchemy default DateTime
1639. Checking if a string can be converted to float in Python
1640. Calculating arithmetic mean (one type of average) in Python
1641. Fastest way to get the first object from a queryset in django?
1642. How to filter a dictionary according to an arbitrary condition function?
1643. Pandas percentage of total with groupby
1644. How to create a temporary directory and get its path/ file name?
1645. Does Python optimize tail recursion?
1646. multiprocessing vs multithreading vs asyncio
1647. Using property() on classmethods
1648. Pandas DataFrame Groupby two columns and get counts
1649. UnboundLocalError on local variable when reassigned after first use
```

Główny program najpierw pobiera już przygotowany zbiór danych a następnie oczekuje na wprowadzenie przez użytkownika pytania w celu sprawdzenia czy takie pytanie już istnieje lub istnieją do niego podobne również pod względem semantycznym.

Po otrzymaniu pytania program przygotowuje tablice krotek które zawierają w sobie już istniejące pytanie jak i wskaźnik podobieństwa wyrażony w postaci liczby między 0 a 1.

W celu dobrania wskaźnika podobieństw do każdego pytania wykorzystaliśmy napisaną przez nas funkcje:

```python
def similarity(string1, string2):
    string1_synsets = [get_synsets(*word) for word in pos_tag(word_tokenize(string1)) if get_synsets(*word)]
    string2_synsets = [get_synsets(*word) for word in pos_tag(word_tokenize(string2)) if get_synsets(*word)]

    similarity_points = 0
    for set1 in string1_synsets:
        max_points = 0
        for set2 in string2_synsets:
            points = set1.path_similarity(set2)
            if points > max_points:
                max_points = points
        similarity_points += max_points

    if len(string1_synsets) == 0:
        return 0

    return similarity_points / len(string1_synsets)


def get_synsets(word, tag):
    wordnet_tag = None

    if tag.startswith('N'):
        wordnet_tag = 'n'
    elif tag.startswith('V'):
        wordnet_tag = 'v'
    elif tag.startswith('J'):
        wordnet_tag = 'a'
    elif tag.startswith('R'):
        wordnet_tag = 'r'
    else:
        return None

    try:
        return wordnet.synsets(word, wordnet_tag)[0]
    except:
        return None
```

Funkcja ta z wykorzystuje worldnet czyli bazę słownikową języka angielskiego stworzoną przez Uniwersytet Princeton. Stosujemy do szukania podobieństwa semantycznego poszczególnych wyrazów zawartych w pytaniu.

Przy użyciu pierwszych linijek tworzymy dwie tablice zawierające już przetworzone słowa z pytania użytkownika oraz pytania z zestawu danych z pominięciem słów, które nie wpływają na znaczenie semantyczne pytania (np. “the”, “a”).

```python
def similarity(string1, string2):
    string1_synsets = [get_synsets(*word) for word in pos_tag(word_tokenize(string1)) if get_synsets(*word)]
    string2_synsets = [get_synsets(*word) for word in pos_tag(word_tokenize(string2)) if get_synsets(*word)]
```

Po przygotowaniu takich tablic za pomocą funkcji path_similarity() porównujemy ze sobą po kolei każdy wyraz z pytania użytkownika z każdym wyrazem pytania z zestawu i szukamy wyniku najwyższego po czym dodajemy wynik każdego wyrazu i uśredniamy.

```
    similarity_points = 0
    for set1 in string1_synsets:
        max_points = 0
        for set2 in string2_synsets:
            points = set1.path_similarity(set2)
            if points > max_points:
                max_points = points
        similarity_points += max_points

    if len(string1_synsets) == 0:
        return 0

    return similarity_points / len(string1_synsets)
```

Mając już tablice krotek z pytaniami oraz ich wskaźnikiem podobieństwa sortujemy ją od wartości najwyższych do najniższych. Następnie bierzemy 4 najwyższe wyniki o ile posiadają wskaźnik podobieństwa na poziomie przynajmniej 0.75 w celu otrzymania tylko faktycznie podobnych pytań i zwracamy te pytania w formie listy.

```
    similarity_list.sort(key=lambda x: x[1], reverse=True)

    for j in range(0, 4):
        if similarity_list[j][1] > 0.75:
            result_questions.append(similarity_list[j][0])
```

Posiadając już tak przygotowaną tablice, jesteśmy w stanie wyświetlić te najlepsze wyniki dla użytkownika.

3. Podsumowanie i sprawdzenie działania

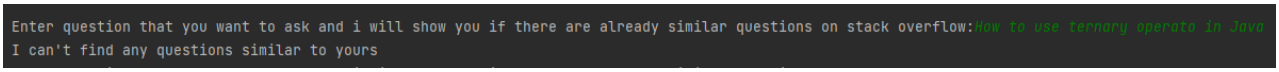
Poniżej przykład przy zadaniu pytania “How to use ternary operator in Pyhon?”

```
Enter question that you want to ask and i will show you if there are already similar questions on stack overflow:How to use ternary operator in Pyhon?
List of similar questions:
1)  Does Python have a ternary conditional operator?
```

Jak widać, narzędzie to odnalazło pytanie o bardzo podobnym znaczeniu, mimo że zostało ono zapisane w widocznie różny sposób (Dodatkowo

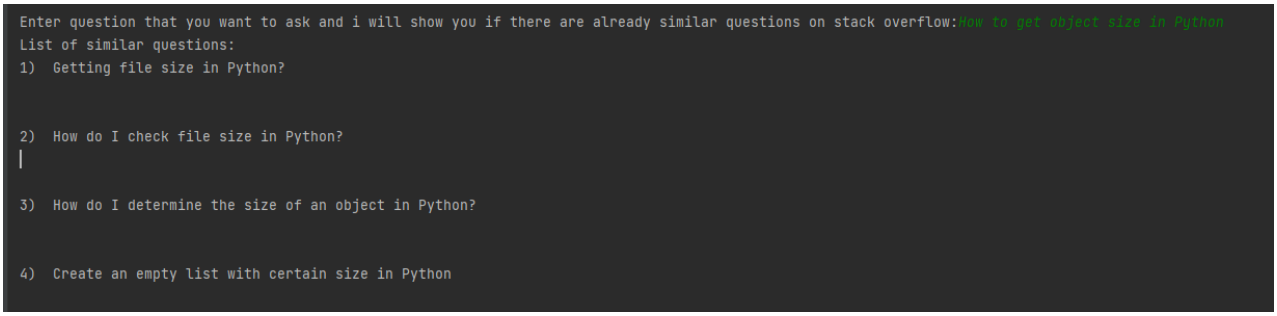
pojawiła się w pytaniu literówka która udowadnia, że program radzi sobie nawet z drobnymi literówkami). Wypisane zostało tylko jedno proponowane pytanie co oznacza, że w naszym zbiorze tylko to jedno pytanie miało wskaźnik podobieństwa na wymaganym poziomie.

Kolejny przykład tym razem ze zmianą języka programowania, którego dotyczy pytanie z Python na Java:



Na tym przykładzie widać, że mimo iż pytanie było sformułowane w bardzo podobny sposób co poprzednie to różnica wywołana zmianą nazwy języka programowania była wystarczająco duża, aby program nie zaproponował ponownie pytania co w poprzednim przykładzie.

Poniżej widzimy przykład dla pytania “How to get object size in Python”



Jak widać, tym razem otrzymaliśmy 4, czyli maksymalną ilość pytań podobnych, aczkolwiek nie wszystkie są już aż tak podobne. Pierwsze jak i drugie zaproponowane przez program pytania są skonstruowane bardzo podobnie, ale dotyczą różnych rzeczy (użytkownik zapytał się o rozmiar obiektu a otrzymał rozmiar pliku). Trzecie pytanie za to jest już bardzo dobrze dopasowane pod względem semantycznym. Ostatnie pytanie zaproponowane przez nasze narzędzie nie jest już aż tak podobne do zadanego pytania jak poprzednie, rozpoznał podobieństwo związane z językiem Python i tym, że chodzi nam o wielkość czegoś, jednakże użytkownikowi chodziło o rozmiar obiektu a nie listy i nie o to jak stworzyć go z określonym rozmiarem tylko jak ten rozmiar otrzymać.

Przykład ten pokazuje, że nasze narzędzie pomimo dość dużej skuteczności, wciąż jest dalekie od perfekcji. W celu osiągniecia lepszych wyników, należałoby między innymi zarejestrować aplikacje w stock exchange w celu rozbudowy zbioru danych oraz dopracowanie funkcji przypisującej punkty, aby poprawić jej skuteczność.

4. Źródła:

- https://api.stackexchange.com/docs
- https://www.kaggle.com/c/predict-closed-questions-on-stack-overflow/data
- https://www.nltk.org/howto/wordnet.html
- https://wordnet.princeton.edu/
- https://www.nltk.org/api/nltk.tag.html