

Wyszukiwanie podobnych pytań na StackOverflow

Wykonali:
Marcel Czurczak
Igor Bobek
Dominik Dziuba

1. Abstrakt.....	3
2. Wstęp.....	4
2.1. Cel.....	4
2.2. Zakres.....	4
2.3. Metodyka.....	4
3. Część teoretyczna.....	5
4. Część praktyczna.....	6
4.1. Algorytm wykorzystujący analizę dopasowania.....	7
4.2. Algorytm wykorzystujący analizę głównych składowych (PCA).....	9
5. Podsumowanie.....	11
6. Bibliografia.....	12

1. Abstrakt

Praca ta koncentruje się na opracowaniu i implementacji efektywnego mechanizmu wyszukiwania podobnych pytań w platformie Stack Overflow. W kontekście dynamicznej i rozwijającej się społeczności programistycznej, problem identyfikacji istniejących pytań o podobnej tematyce staje się kluczowym wyzwaniem. W ramach tego badania analizowane są różne podejścia do pomiaru podobieństwa treści pytań, wykorzystując zaawansowane metody przetwarzania języka naturalnego m.in. analizę głównych składowych oraz analizę dopasowania.

2. Wstęp

2.1. Cel

Celem projektu było stworzenie narzędzia służącego do wyszukiwania podobnych pytań z serwisu StackOverflow na podstawie pytania zadawanego przez użytkownika.

2.2. Zakres

Zakres pracy obejmował:

1. Pobranie danych z serwisu Kaggle, które zawierają wszystkie informacje o postach z serwisu StackOverflow.
2. Pobranie tytułów postów i zapisanie ich w osobnym pliku.
3. Stworzenie programu do analizy tekstu, wykorzystując analizę dopasowania oraz analizę semantyczną.
4. Prezentacja wyników poszczególnych metod oraz porównanie ich.

2.3. Metodyka

Narzędzia użyte w projekcie:

- python 3.10,
- numpy,
- pandas,
- scikit-learn,
- nltk

3. Część teoretyczna

Tokenizacja [1] to procedura mająca na celu przekształcenie sekwencji znaków w mniejsze jednostki zwane tokenami. Głównym powodem, dla którego ten proces jest istotny, polega na ułatwieniu maszynom zrozumienia ludzkiego języka poprzez podział go na bardziej zrozumiałe fragmenty. Dzięki podziałowi tekstu na tokeny, analiza staje się bardziej efektywna, umożliwiając maszynom lepsze zrozumienie i interpretację języka naturalnego.

TF-IDF [2] to technika oceny wagi terminów, oparta na analizie ich częstości występowania w danym dokumencie (TF - ang. Term Frequency) oraz ich rozkładu w całym zbiorze dokumentów (IDF - ang. Inverse Document Frequency). Wartości TF-IDF nadają większą wagę słowom, które pojawiają się rzadko w korpusie, gdyż posiadają one większą siłę dyskryminacyjną. W praktyce TF-IDF pomaga zidentyfikować i wyróżnić istotne terminy charakteryzujące konkretny tekst w kontekście całego zbioru dokumentów.

Analiza głównych składowych (ang. Principal Component Analysis) [3] to metoda analizy czynnikowej wykorzystywana w statystyce. Zbiór danych składających się z N obserwacji, z każdą zawierającą K zmiennych, można interpretować jako chmurę N punktów w przestrzeni K -wymiarowej. Celem tej metody jest obrót układu współrzędnych w taki sposób, aby pierwsza współrzędna miała maksymalną wariancję, a następnie kolejne współrzędne były ułożone malejąco pod względem wariancji. Umożliwia to redukcję wymiarów danych, co ułatwia analizę i interpretację jednocześnie zachowując istotne informacje z pierwotnego zbioru danych. Otrzymane składowe główne mogą być interpretowane jako nowe, nieskorelowane ze sobą zmienne, które prezentują główne kierunki zmienności danych.

Podobieństwo cosinusowe (ang. Cosine similarity) [4] to miara stopnia podobieństwa między dwoma niezerowymi wektorami w przestrzeni iloczynu wewnętrznego. Wykorzystuje kąt cosinusowy między tymi wektorami do określenia, na ile są one skierowane w tym samym lub podobnym kierunku. Owa miara jest powszechnie wykorzystywana w analizie tekstu do oceny podobieństwa pomiędzy dokumentacją.

4. Część praktyczna

Do stworzenia modeli wyszukiwania konieczne było zaimportowanie odpowiednich bibliotek.

```
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.decomposition import PCA
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

Rys. 1 - Kod odpowiedzialny za importowanie bibliotek

Plik *train.csv* pobrany z serwisu Kaggle, zawierał dane dotyczące postów dostępnych na serwisie StackOverflow. Funkcja przedstawiona na rys. 2 pobiera wszystkie tytuły znajdujące się w kolumnie “Title” i zapisuje je do pliku *titles_output.csv*.

```
def extract_titles(input_file, output_file):
    df = pd.read_csv(input_file, chunksize=10000)

    titles = []

    for chunk in df:
        titles.extend(chunk['Title'].tolist())

    pd.DataFrame({'Title': titles}).to_csv(output_file, index=False)

extract_titles('train.csv', 'titles_output.csv')
```

Rys. 2 - Funkcja pobierająca tytuły z pliku *train.csv*

Na rys. 3 przedstawiony kod korzysta z narzędzi pakietu Natural Language Toolkit (NLTK). Lista ‘stopwords’ zawiera listę słów nie przenoszących żadnej istotnej informacji (a, an, the w języku angielskim). Dane ‘punkt’ to dane służące do tokenizacji tekstu.

```

nltk.download('stopwords')
nltk.download('punkt')

stop_words = set(stopwords.words('english'))

```

Rys. 3 - pobieranie danych z pakietu NLTK

W pliku *titles_output.csv* znajduje się około 3,5 mln. tytułów. Niestety program nie był w stanie przetworzyć tak dużej liczby danych, więc należało losowo wybrać rekordy i zapisać je do pliku *titles.csv*

```

import pandas as pd

df = pd.read_csv('titles_output.csv')

random_rows = df.sample(n=1000, random_state=42)

random_rows.to_csv('titles.csv', index=False)

```

Rys. 4 - Fragment kodu odpowiedzialny za losowe wybieranie rekordów do testów

Poniższy fragment kodu pokazuje pobranie wyżej wymienionych danych przy użyciu pakietu *pandas*.

```

df = pd.read_csv('titles.csv', header=None, names=['Title'])
titles = df['Title'].tolist()

```

Rys. 5 - Fragment kodu odpowiedzialny za pobranie tytułów z pliku *titles.csv*

4.1. Algorytm wykorzystujący analizę dopasowania

Kod znajdujący się na rys 7. odpowiada za stworzenie macierzy TF-IDF. Początkowo dane są wektoryzowane przy pomocy klasy *TfidfVectorizer*, gdzie w konstruktorze przekazywana jest funkcja *lambda*, służąca do tokenizacji tekstu przy pomocy funkcji *word_tokenize* z biblioteki NLTK. Te dane następnie przekazywane do funkcji *fit_transform*, która przekształca je do macierzy TF-IDF. Ta macierz jest następnie przekształcana do macierzy *numpy* przy pomocy funkcji *toarray()*.

```

tfidf = TfidfVectorizer(tokenizer=lambda text: [word for word in word_tokenize(text) if word.lower() not in stop_words],
                        dtype=np.float32)
X = tfidf.fit_transform(raw_documents=titles).toarray()

```

Rys. 7 - Fragment kodu odpowiedzialny za wektoryzację danych wejściowych

Rys. 8 przedstawia fragment kodu odpowiedzialny za pobranie pytania przez użytkownika, przekształcenie go przy użyciu modelu TF-IDF, obliczenie wyników korzystając z funkcji *cosine_similarity* oraz wyświetlenie ich. Wyniki są filtrowane na podstawie zmiennej *threshold*, aby wyświetlane były tylko te najbardziej podobne pytania.

```
while True:
    question = input("Ask something: ")

    if question == "exit":
        break

    Q = tfidf.transform([question]).toarray()
    Q_vec = Q[0, :]
    X_vec = np.array(X)

    similarities = cosine_similarity([Q_vec], X_vec)

    threshold = 0.4

    filtered_indices = np.where(similarities[0] > threshold)[0]

    sorted_indices = filtered_indices[np.argsort(similarities[0, filtered_indices])[:-1]]

    print(f"New question: {question}")
    print("Related questions (sorted by similarity):")
    for idx in sorted_indices:
        print(f"Similarity score: {similarities[0, idx]} - {titles[idx]}")
```

Rys. 8 - Fragment kodu odpowiedzialny za znalezienie podobnych zapytań na podstawie pytania użytkownika

Na rysunkach 9, 10 oraz 11 przedstawione zostały wyniki wykorzystania analizy dopasowania.

```
New question: execute sql commands in powershell
Related questions (sorted by similarity):
Similarity score: 0.6239813566207886 - Execute SQL*Plus from PowerShell
Similarity score: 0.4145209789276123 - How to execute PowerShell scripts in C# using the Task Parallel Library
```

Rys. 9 - Wynik dla analizy dopasowania dla pytania "execute sql commands in powershell"

```
New question: how to comment unit tests
Related questions (sorted by similarity):
Similarity score: 0.8257208466529846 - Should I comment my tests?
Similarity score: 0.564498245716095 - delegate methods for unit tests
Similarity score: 0.5423920154571533 - Giving character to your unit tests
Similarity score: 0.48968806862831116 - Can unit tests have a execution time criteria?
```

Rys. 10 - Wynik dla analizy dopasowania dla pytania "how to comment unit tests"


```
New question: warnings in php
Related questions (sorted by similarity):
Similarity score: 0.7082716226577759 - PHP set warnings as fatal
Similarity score: 0.4884476363658905 - php help needed, blogger api / xml warnings??
Similarity score: 0.46406757831573486 - Compiler warnings in others' libraries
```

Rys. 10 - Wynik dla analizy dopasowania dla pytania "warnings in php"

4.2. Algorytm wykorzystujący analizę głównych składowych (PCA)

Na rys. 10 kod przedstawia tworzenie macierzy TF-IDF podobnie jak na rys. 7. W tym przypadku dane przed przekazaniem ich do filtrowania są redukowane przy pomocy algorytmu analizy głównych składowych. Następuje inicjalizacja obiektu PCA przy pomocy 1000 komponentów, co oznacza, że zostanie zachowanych tylko 1000 najważniejszych komponentów składowych podczas redukcji wymiarów.

```
tfidf = TfidfVectorizer(tokenizer=lambda text: [word for word in word_tokenize(text) if word.lower() not in stop_words],
                        dtype=np.float32)

tfidf_docs = tfidf.fit_transform(raw_documents=titles).toarray()

pca = PCA(n_components=1000)
pca_result = pca.fit_transform(tfidf_docs)
```

Rys. 12 - Fragment kodu odpowiedzialny za wektoryzację oraz redukcję wymiarowości tych wektorów za pomocą algorytmu PCA.

Na rys. 11 przedstawiono fragment kodu, który pobiera pytanie od użytkownika, przekształca je przy pomocy wcześniej przygotowanego modelu TF-IDF. Następnie wektor reprezentujący nowe pytanie jest poddawany redukcji wymiarów przy użyciu PCA. Wynik jest obliczany przy użyciu funkcji *cosine_similarity*, a następnie przefiltrowany i wyświetlony, tak jak w przypadku analizy dopasowania.

```
while True:
    new_question = input("Ask something: ")

    if new_question == "exit":
        break

    new_question_vector = tfidf.transform([new_question]).toarray()
    new_question_pca = pca.transform(new_question_vector)

    similarities = cosine_similarity(new_question_pca, pca_result)

    threshold = 0.4

    related_questions_indices = similarities[0] >= threshold

    sorted_indices = sorted(range(len(similarities[0])), key=lambda k: similarities[0][k], reverse=True)

    print(f"New question: {new_question}")
    print("Related questions (sorted by similarity):")
    for idx in sorted_indices:
        if similarities[0][idx] >= threshold:
            print(f"Similarity Score: {similarities[0][idx]} - {titles[idx]}")
```

Rys. 13 - Fragment kodu odpowiedzialny za analizę podobnych pytań na podstawie TF-IDF oraz PCA.

Na rysunkach 14, 15 oraz 16 przedstawione zostały wyniki wykorzystania analizy dopasowania.

```
New question: execute sql commands in powershell
Related questions (sorted by similarity):
Similarity Score: 0.7506935000419617 - Execute SQL*Plus from PowerShell
Similarity Score: 0.5488085150718689 - Open gnome terminal programmatically and execute commands after bashrc was executed
Similarity Score: 0.48244404792785645 - How to execute PowerShell scripts in C# using the Task Parallel Library
Similarity Score: 0.4583875834941864 - How to execute text as PHP
Similarity Score: 0.43924248218536377 - Execute python Script on Crontab
Similarity Score: 0.42394277453422546 - How do I execute data in Webkit consoles by using javascript?
Similarity Score: 0.41536927223205566 - How to execute transition on value change in h:selectOneMenu ?
```

Rys. 14 - Wynik analizy głównych składowych dla pytania "execute sql commands in powershell"

```
New question: how to comment unit tests
Related questions (sorted by similarity):
Similarity Score: 0.8056601881980896 - Should I comment my tests?
Similarity Score: 0.7713750600814819 - Giving character to your unit tests
Similarity Score: 0.7502776980400085 - Can unit tests have a execution time criteria?
Similarity Score: 0.6673493385314941 - delegate methods for unit tests
Similarity Score: 0.4984351694583893 - Selenium tests, how would you do?
Similarity Score: 0.4170316457748413 - Multithreaded Unit Testing
Similarity Score: 0.40908676385879517 - How to get the authentication status in clearance for tests
Similarity Score: 0.4005139470100403 - BlazeDS error when running Flexunit tests: Detected duplicate HTTP-based FlexSessions
```

Rys. 15 - Wynik dla analizy dopasowania dla pytania "how to comment unit tests"

```
New question: warnings in php
Related questions (sorted by similarity):
Similarity Score: 0.7742890119552612 - Why is locking such a mess in PHP?
Similarity Score: 0.6721699237823486 - PHP set warnings as fatal
Similarity Score: 0.6076607704162598 - preg_match differences in PHP 5.3.2 vs. PHP 5.2.1
Similarity Score: 0.5982716083526611 - php cli include_once error
Similarity Score: 0.5927965044975281 - PHP MYSQL OPTIMIZATION
Similarity Score: 0.586287260055542 - PHP - managing timezones
Similarity Score: 0.584235668182373 - Synchronize Firebird with MySQL in PHP
Similarity Score: 0.5668741464614868 - verify a rsa sign from java in php
Similarity Score: 0.565962553024292 - Openinviter not importing contacts of Hotmail using php
Similarity Score: 0.5200716853141785 - workaround for multiple inheritences in PHP?
Similarity Score: 0.5115646123886108 - Securely using the PHP exec function
Similarity Score: 0.5102816820144653 - PHP OO Call to a non-object
Similarity Score: 0.5041036009788513 - HTTP if-none-match and if-modified-since and 304 clarification in PHP
Similarity Score: 0.49780455231666565 - PHP date of birth checker
Similarity Score: 0.48069247603416443 - PHP Link in echo
Similarity Score: 0.47438305616378784 - PHP logic if statments
Similarity Score: 0.47077852487564087 - Redis and PHP (Rediska) intersect on set
Similarity Score: 0.46584591269493103 - Do native _SEPARATOR in PHP
Similarity Score: 0.46223020553588867 - PHP pages inclusion
Similarity Score: 0.45815420150756836 - OLE_COLOR conversion in PHP
Similarity Score: 0.4570351541042328 - PHP captcha - help
Similarity Score: 0.453952819108963 - install php on server (iis6) when php already exists due to backup exec
Similarity Score: 0.4494573473930359 - In PHP, How would I use a variable that was created in another php page?
Similarity Score: 0.4491172730922699 - online exam in php
Similarity Score: 0.44374293088912964 - PHP List Explode Variable
```

Rys. 16 - Wynik analizy głównych składowych dla pytania "warnings in php"

5. Podsumowanie

Udało się zrealizować założony cel. W trakcie jego realizacji okazało się, że metoda analizy głównych składowych daje bardziej wiarygodne wyniki niż analiza dopasowania.

6. Bibliografia

- [1] <https://www.datacamp.com/blog/what-is-tokenization>
- [2] <http://gatak.pl/2021/01/28/scikit-learn-tf-idf>
- [3] <https://medium.com/intro-to-artificial-intelligence/principal-component-analysis-pca-cd282196b7d5>
- [4] <https://fineproxy.org/pl/wiki/cosine-similarity>