

# Narzędzie do tworzenia podsumowań tekstu

MS, GZ

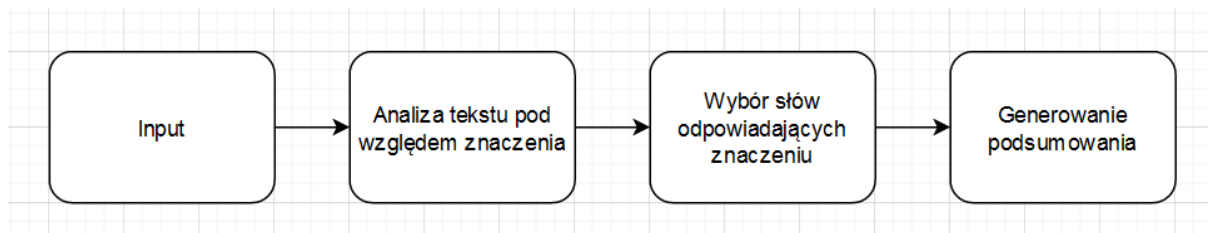
## 1. Abstrakt:

Implementacja skryptu pozwalającego na wygenerowanie podsumowania artykułu podanego przez użytkownika. Artykuł pobierany jest na podstawie podanego przez użytkownika linku. Następnie skrypt wylicza podobieństwo zdań i na tej podstawie wybiera zdania o największym znaczeniu, które tworzą podsumowanie.

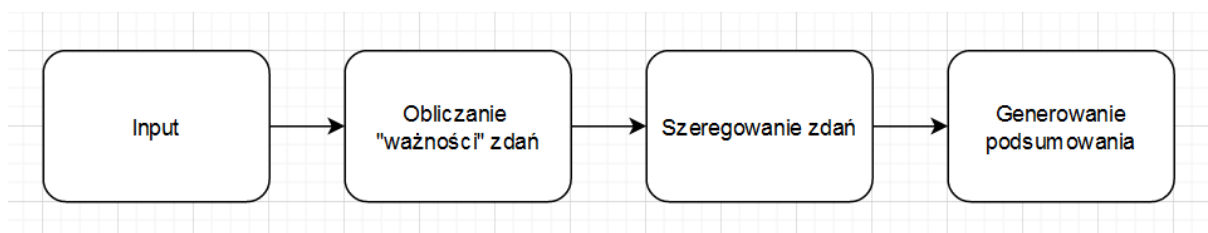
## 2. Wstęp:

Istnieją dwa podejścia do generowania podsumowań: abstrakcyjne i ekstrakcyjne [5].

Metody abstrakcyjne - działają podobnie do tego jak ludzie podsumowują tekst. Metoda polega na wyprodukowaniu nowego, krótszego tekstu zawierającego najważniejsze informacje. Słowa podsumowania wybierane są na podstawie ich znaczenia, nawet jeżeli nie koniecznie pojawiają się w dokumencie źródłowym. Wymaga to zastosowania zaawansowanych technik języka naturalnego do interpretacji tekstu.



Metody ekstrakcyjne - polegają na wybraniu z tekstu takiego zbioru zdań, które zachowują najważniejsze informacje. Na podstawie obranego algorytmu szereguje się zdania według ich ważności i najistotniejsze łączy się w podsumowanie



### 3. Implementacja:

Zdecydowaliśmy się wykorzystać podejście ekstrakcyjne, ponieważ abstrakcyjne metody podsumowujące muszą radzić sobie z tak skomplikowanymi problemami jak reprezentacja semantyczna, wnioskowanie i generowanie języka naturalnego.

Do implementacji rozwiązania wykorzystaliśmy biblioteki:

- NumPy - podstawowy pakiet do obliczeń naukowych w Pythonie [1]
- NetworkX - pakiet do tworzenia, manipulowania i badania struktury, dynamiki i funkcji złożonych sieci w Pythonie [2]
- Sklearn - biblioteka do uczenia maszynowego w Pythonie. Zapewnia wybór wydajnych narzędzi do uczenia maszynowego i modelowania statystycznego [3]
- newspaper3k - newspaper3k to biblioteka Pythona używana do scrapowania artykułów internetowych [4]

#### 3.1. Wczytywanie pliku

Artykuły scrapowane są na podstawie URL za pomocą biblioteki newspaper3k. [4] Treść artykułu jest pobierana, parsowana a następnie wyodrębniany jest sam tekst.

```
from newspaper import Article
import textwrap as tr

while(True):
    url = input()
    if not url:
        break
    article = Article(url)
    print("Reading article...")
    article.download()
    article.parse()
    print("Summary for article \"" + article.title + "\":\n")
    summary = getSummary(article.text)
    print(tr.fill(summary, width=100))
```

#### 3.2. Podział na zdania

Aby móc wygenerować podsumowanie najpierw dzielimy tekst na zdania. Służy do tego metoda getSentences, która za argument wywołania przyjmuje tekst artykułu.

```
def getSentences(lines):
    lines = lines.replace('\n', ' ').replace('\t', '')
```

```

sentences = lines.split(". ")
originalSentences = [s + "." for s in sentences]
sentences = [s.strip().lower().replace("[^a-zA-Z]", " ") for s in
sentences]
return sentences, originalSentences

```

Metoda zwraca dwie wartości: `sentences`, która jest listą list słów, pozbawioną znaków specjalnych, liczb i ujednoczoną do małych liter oraz `originalSentences`, która zawiera listę niezmiennych zdań.

### 3.3. Obliczanie podobieństwa

Obliczanie wartości podobieństwa zdań odbywa się w metodzie `sentenceSimilarity`.

```

def sentenceSimilarity(sentence_vec1, sentence_vec2):
    return cosine_similarity(sentence_vec1.reshape(1, -1),
sentence_vec2.reshape(1, -1))

```

Wykorzystuje ona metodę `cosine_similarity` z pakietu `sklearn`, która przyjmuje za argumenty dwa zwektoryzowane zdania i zwraca ich podobieństwo cosinusowe. [3]

### 3.4. Tworzenie macierzy podobieństwa

Aby zidentyfikować najważniejsze zdania w tekście, tworzona jest macierz ich podobieństwa cosinusowego. Zajmuje się tym metoda `sentenceSimilarityMatrix`.

```

def sentenceSimilarityMatrix(sentences):
    tfidfvectorizer = TfidfVectorizer(analyzer='word',
stop_words="english")
    sentences_vec =
np.array(tfidfvectorizer.fit_transform(sentences).todense().copy())
    similarity_matrix = np.zeros((len(sentences), len(sentences)))

    for i in range(len(sentences)):
        for j in range(len(sentences)):
            if i == j:
                continue
            similarity_matrix[i][j] = sentenceSimilarity(sentences_vec[i],
sentences_vec[j])
    return similarity_matrix

```

Używa ona TfidfVectorizera do usunięcia słów stopu i stworzenia gęstych wektorów. [3]  
Następnie porównuje wszystkie zdania ze sobą nawzajem i zwraca macierz wyników.

### 3.5. Generowanie podsumowania

Główną metodą skryptu jest metoda `getSummary`, która generuje podsumowanie.

```
def getSummary(article):
    parsedSentences, originalSentences = getSentences(article)
    sentence_matrix = sentenceSimilarityMatrix(parsedSentences)

    sentence_similarity_graph = nx.from_numpy_array(sentence_matrix)
    scores = nx.pagerank(sentence_similarity_graph)
    ranked_sentence = sorted(((scores[i],s) for i,s in
enumerate(originalSentences)), reverse=True)

    summary = ""
    for i in range(0,5):
        summary = summary + ranked_sentence[i][1]

    return summary
```

Z pomocą metod opisanych wyżej oblicza macierz podobieństwa cosinusowego. Następnie używa metody `from_numpy_array` z pakietu `networkx`, która na podstawie macierzy tworzy graf. Metoda `pagerank` z pakietu `networkx` oblicza ranking węzłów na grafie za pomocą algorytmu `pagerank`. [2]

Podsumowanie tworzone jest ze zdań o największej wartości podobieństwa cosinusowego.

## 4. Obsługa programu

Program po uruchomieniu wyświetla pole do wpisania tekstu. W tym polu należy umieścić link do artykułu, dla którego ma zostać wygenerowane podsumowanie. Po wpisaniu linku i wciśnięciu `Enter` program wczyta artykuł i wyświetli jego podsumowanie:

```
... https://news.berkeley.edu/2022/11/22/massive-traffic-experiment-pits-machine-learning-against-phantom-jams/
Reading article...
Summary for article "Massive traffic experiment pits machine learning against 'phantom' jams":

"With this experiment, we hope to better understand the impact of these systems, and also make sure
that whatever the impact is, it benefits traffic overall and not just individual vehicles."
Creating "socially acceptable" AI As part of the CIRCLES consortium, UC Berkeley researchers have
taken the lead in developing the machine learning algorithms that govern how fast AI-powered
vehicles should go. Department of Energy (DOE) in 2020, the CIRCLES team began preparations to repeat
the experiment on a much larger scale, this time integrating the AI-equipped vehicles into the
normal flow of highway traffic. "That's precisely what AI technology is able to fix - it can direct
the vehicle to things that are not intuitive to humans, but are overall more efficient." Bayen is
principal investigator of the CIRCLES Consortium, a multi-university research collaboration
dedicated to using machine learning to improve traffic flow and increase energy efficiency. Each
vehicle was equipped with an AI-powered cruise control system designed to automatically adjust the
speed of the vehicle to improve the overall flow of traffic - essentially turning each car into its
own "robot traffic manager." "Driving is very intuitive. In addition to adjusting the speed of the
vehicle in response to local conditions, the technology also incorporates information about traffic
conditions and adjusts the speed to help smooth the overall flow of traffic."

```

Po wyświetleniu podsumowania można wkleić następną link lub wcisnąć Enter aby program zakończył działanie.

## 5. Podsumowanie:

Program generuje podsumowanie celnie oddające treść artykułu. Ze względu na wykorzystanie metody ekstrakcyjnej jego działanie jest relatywnie szybkie.

## 6. Bibliografia

1. NumPy documentation - <https://numpy.org/doc/stable/>. Data dostępu: 11.24.2022
2. NetworkX - <https://networkx.org/documentation/stable/index.html>. Data dostępu: 11.24.2022
3. Scikit-learn - <https://scikit-learn.org/stable/>. Data dostępu 11.24.2022
4. Newspaper3k - <https://newspaper.readthedocs.io/en/latest/> . Data dostępu: 11.24.2022
5. Towards Data Science - <https://towardsdatascience.com/understand-text-summarization-and-create-your-own-summarizer-in-python-b26a9f09fc70> Data dostępu: 11.24.2022