

NARZĘDZIE DO TWORZENIA PODSUMOWAŃ TEKSTU

Michał Cichocki

Adrian Cygan

Szymon Czajkowski

semestr 7, Informatyka, Wydział Informatyki i Telekomunikacji

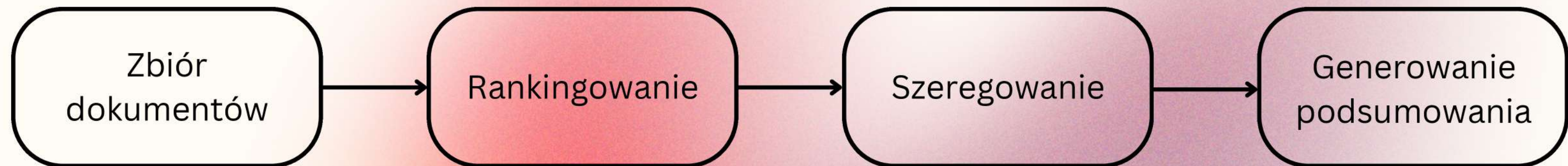
Wymagania funkcjonalne

- Pobieranie i ekstrakcja treści artykułu podanego poprzez link.
- Wstępne przetwarzanie otrzymanego tekstu
- Generowanie podsumowania artykułu w ilości zdań podanych w parametrze.
- Możliwość wyboru techniki wektoryzacji zdań.
- Narzędzie ma być udostępnione w formie paczki języka Python

Część Teoretyczna

Metody ekstrakcyjne

Polegają na wybieraniu ze zbioru zdań takiego podzbioru, który reprezentuje najważniejsze punkty tekstu



Nie generują nowego tekstu

Część Teoretyczna



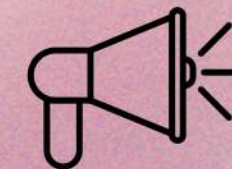
Scrapping

Proces automatycznego pobierania i przetwarzania danych ze strony internetowej.



Tokenizacja

Konwertowanie danych wejściowych na tokeny. Dokumenty są dzielone zdania, a zdania na słowa, liczby lub inne znaki.

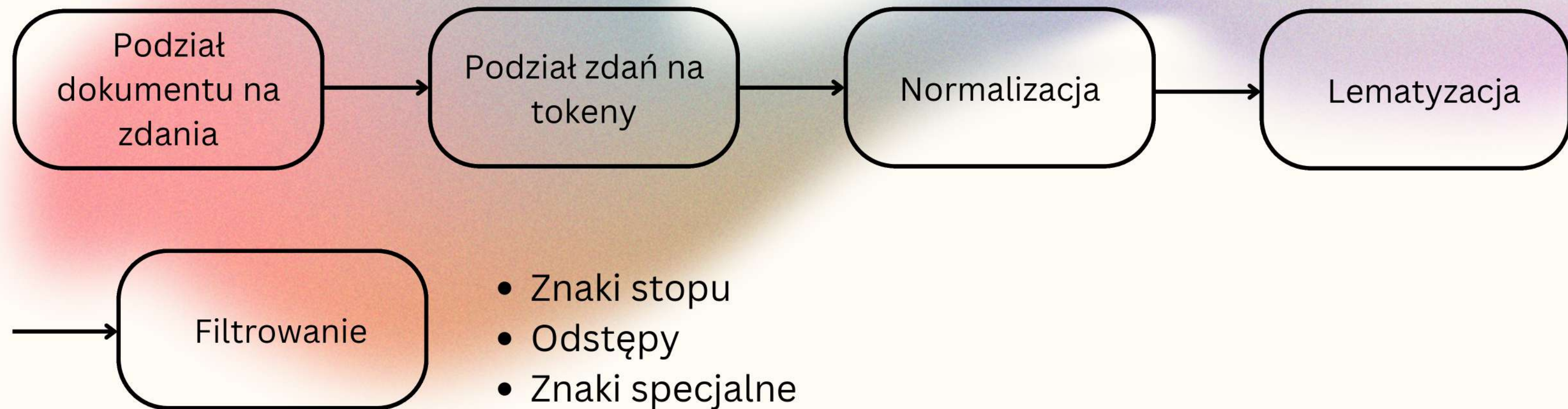


Lematyzacja

Wyodrębnienie z odmienionego słowa jego formę bazową

Część Teoretyczna

Tokenizacja i wstępne przetwarzanie- działanie



Część Teoretyczna

Techniki wektoryzacji

Worek słów

Dokument można przedstawić jako wektor, gdzie każdy element oznacza częstotliwość występowania słowa, pomijając kolejność słów.

TF-IDF

TF-IDF ocenia znaczenie słowa w dokumencie, uwzględniając jego częstość i istotność w całym zbiorze dokumentów.

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

Część Teoretyczna

Podobieństwo cosinusowe

Podobieństwo cosinusowe mierzy, jak dwa teksty, przedstawione jako wektory liczbowe, są do siebie podobne w przestrzeni wielowymiarowej.

$$\cos(\theta) = \frac{A \bullet B}{\|A\| \|B\|}$$

Page Rank

Algorytm PageRank oblicza wagi węzłów w grafie na podstawie liczby krawędzi i wag węzłów, z których pochodzą.

Część praktyczna

Wykorzystane narzędzia:

- NLTK
- Networkx
- scikit-learn
- Morfeusz
- BeautifulSoup
- Inne narzędzia

Część praktyczna

Etapy działania:

Scrapping

```
def scrape_text_from_url(self) -> tuple[string, string, list[string]]: 1 usage
    response = requests.get(self.url_path)
    if response.status_code == 200:
        soup = BeautifulSoup(response.content, features='html.parser')
        h1_title = soup.find(name='h1', class_='title').get_text() if soup.find(name='h1', class_='title') else 'No title found'
        p_lead = soup.find(name='p', class_='lead').get_text() if soup.find(name='p', class_='lead') else 'No lead found'
        div_content_parts = soup.find(name='div', class_='contentparts')
        p_content_parts = []
        if div_content_parts:
            for p in div_content_parts.find_all(name='p', class_='contentpart--text'):
                text = p.get_text()
                if '\xa0' not in text and 'ZOBACZ WIDEO' not in text and not (p.find('a') and text == p.find('a').get_text()):
                    p_content_parts.append(text)
        return h1_title, p_lead, p_content_parts
    else:
        raise SummarizerException(f"Failed to fetch data from {self.url_path}")
```


Część praktyczna

Etapy działania:

Tokenizacja i wstępne przetwarzanie

```
original_sentences = raw_text.split(". ")

processed_sentences = [self.preprocess_text(sentence).split(" ") for sentence in original_sentences if sentence.strip()]
```

```
def preprocess_text(self, text, lemmatize=True) -> string: 1 usage
    morf = Morfeusz()
    text = text.lower()
    text = re.sub(pattern: r"\s+", repl: " ", text).strip()
    text = re.sub(pattern: r"[(,)]", repl: "", text).strip()

    if lemmatize:
        lemmatized = [self.find_first_match(word, morf.analyse(word)) for word in text.split()]
        return " ".join(lemmatized)
    return text
```


Część praktyczna

Etapy działania:

Wektoryzacja

```
def similarity_matrix_bow(self, sentences: list[list[str]]): 1 usage
    similarity_matrix = np.zeros((len(sentences), len(sentences)))

    for i in range(len(sentences)):
        for j in range(len(sentences)):
            if i != j:
                similarity_matrix[i][j] = self.sentence_similarity(sentences[i], sentences[j])

    return similarity_matrix

def similarity_matrix_tfidf(self, sentences: list[list[str]]) -> list[list[float]]: 1 usage
    stop_words = self.get_polish_stopwords()

    sentence_texts = [" ".join(sentence) for sentence in sentences]

    tfidf_vectorizer = TfidfVectorizer(stop_words=stop_words)
    tfidf_matrix = tfidf_vectorizer.fit_transform(sentence_texts)

    similarity_matrix = (tfidf_matrix * tfidf_matrix.T).toarray()
    return similarity_matrix
```


Część praktyczna

Etapy działania:

Ranking

```
· sentence_similarity_graph = nx.from_numpy_array(sentence_similarity_matrix)  
· scores = nx.pagerank(sentence_similarity_graph)
```


Część praktyczna

Etapy działania:

Generowanie podsumowania

```
ranked_sentences = sorted(((scores[i], i) for i in range(len(processed_sentences))), reverse=True)

num_sentences = min(self.top_n, len(ranked_sentences))

summarize_text = [original_sentences[ranked_sentences[i][1]] for i in range(num_sentences)]
```


Część praktyczna

Wyniki:

Działanie algorytmu sprawdzono na artykułach sportowych pochodzących z witryny sportowefakty.wp.pl.

Worek słów

- Jeszcze niedawno bał się latać, ale przełamał strach i jest w stanie niedługo stanąć na podium Pucharu Świata
- mówi Adam Małysz o Pawle Wąsku. Poradził sobie z tym i jest w stanie zrobić jeszcze taki krok, by niedługo stawać na podium Pucharu Świata - powiedział po niedzielnych zawodach w Wiśle Adam Małysz. Małysz wskazał nowego lidera polskiej kadry. Będziemy rozmawiać, co zrobić, by pomóc tym najbardziej doświadczonym - zapewnił Małysz

TF-IDF

- Jeszcze niedawno bał się latać, ale przełamał strach i jest w stanie niedługo stanąć na podium Pucharu Świata
- mówi Adam Małysz o Pawle Wąsku. Poradził sobie z tym i jest w stanie zrobić jeszcze taki krok, by niedługo stawać na podium Pucharu Świata - powiedział po niedzielnych zawodach w Wiśle Adam Małysz. Małysz wskazał nowego lidera polskiej kadry. - Paweł ma olbrzymi potencjał

Pytania i Odpowiedzi



Dziękujemy za uwagę

Bibliografia

Witold Kieraś, Marcin Woliński. Morfeusz 2 – analizator i generator fleksyjny dla języka polskiego. Język Polski, XCVII(1):75–83, 2017.

Towards Data Science. (n.d.). Understand text summarization and create your own summarizer in Python. Źródło: <https://towardsdatascience.com/understand-text-summarization-and-create-your-own-summarizer-in-python-b26a9f09fc70>

EITCA Academy. (n.d.). W jaki sposób metoda worka słów przekształca słowa w reprezentacje liczbowe? Źródło: <https://pl.eitca.org/sztuczna-inteligencja/eitc-ai-gcml-Google-Cloud-Machine-Learning/do%C5%9Bwiadczenie-w-uczeniu-maszynowym/zbi%C3%B3r-s%C5%82%C3%B3w-przetwarzaj%C4%85cych-j%C4%99zyk-naturalny/przegl%C4%85d-egzamin%C3%B3w-przetwarzanie-j%C4%99zyka-naturalnego-worek-s%C5%82%C3%B3w/w-jaki-spos%C3%B3b-metoda-worka-s%C5%82%C3%B3w-przekszta%C5%82ca-s%C5%82owa-w-reprezentacje-liczbowe/>

<https://github.com/bieli/stopwords/blob/master/polish.stopwords.txt>