

**POLITECHNIKA KRAKOWSKA
IM. TADEUSZA KOŚCIUSZKI
W KRAKOWIE**

Wydział Informatyki i Telekomunikacji

Informatyka

Raport z realizacji tematu pt.

NARZĘDZIE DO TWORZENIA PODSUMOWAŃ TEKSTU

Michał Cichocki

Adrian Cygan

Szymon Czajkowski

Abstrakt

Raport dotyczy zaprojektowania i implementacji narzędzia, które jest w stanie pobrać treść artykułu z internetu i skonstruować jego streszczenie w oparciu o najważniejsze informacje. Do tego celu zastosowano techniki przetwarzania języka naturalnego oraz przeszukiwania witryn internetowych.

Wstęp	4
Cel pracy.....	4
Zakres.....	4
Metodyka	4
Część teoretyczna	5
Scrapping.....	5
Metody ekstrakcyjne	5
Tokenizacja	5
Worek słów	5
TF-IDF	5
Podobieństwo cosinusowe.....	6
PageRank.....	6
Część praktyczna.....	7
Narzędzia	7
Scrapping.....	7
Tokenizacja	7
Wektoryzacja.....	7
Ranking.....	8
Generowanie podsumowania	8
Wyniki	8
Podsumowanie.....	9

Wstęp

Cel pracy

Celem pracy jest zaprojektowanie i zaimplementowanie narzędzia umożliwiającego budowanie podsumowań tekstów.

Zakres

Projekt obejmuje zbudowanie narzędzia realizującego następujące wymagania funkcjonalne:

1. Pobieranie i ekstrakcja treści artykułu podanego poprzez link.
2. Przetwarzanie wstępne otrzymanego tekstu
3. Generowanie podsumowania artykułu w ilości zdań podanych w parametrze.
4. Możliwość wyboru techniki wektoryzacji zdań.

Narzędzie udostępnione ma być w formie paczki języka Python.

Metodyka

Analiza problemu

Analiza problemu obejmowała identyfikację najważniejszych funkcjonalności jakie nasze rozwiązanie ma oferować oraz podział na podproblemy. Pozwoliło to na efektywne zaplanowanie pracy przy implementacji.

Implementacja

Implementacja oparta została o język Python oraz zewnętrzne biblioteki, implementacja obejmowała następujące etapy:

- Stworzenie scrapera, który pobiera i przetwarza dane ze strony internetowej.
- Implementacja metod służących tokenizacji i lematyzacji.
- Stworzenie narzędzia do generowania podsumowania pobranego tekstu.

Narzędzia

W celu realizacji projektu wykorzystano następujące biblioteki języka Python:

- NLTK - zestaw bibliotek i narzędzi do statycznego przetwarzania języka naturalnego, w programie użyty do obliczenia podobieństwa cosinusowego.
- NetworkX - biblioteka przeznaczona do analizy sieci i grafów, w programie został użyty algorytm PageRank do obliczenia wag zdań z tekstu wejściowego.
- scikit-learn - biblioteka przeznaczona do uczenia maszynowego, w programie użyta do stworzenia wektorów TFIDF.
- Morfeusz – lematyzator dla języka polskiego.
- BeautifulSoup – biblioteka przeznaczona do przetwarzania dokumentów HTML i XML, w programie użyta do web scrapingu.
- Inne narzędzia usprawniające pracę

Część teoretyczna

Metody ekstrakcyjne

Metody ekstrakcyjne polegają na wybieraniu ze zbioru zdań takiego podzbioru, który reprezentuje najważniejsze punkty tekstu. Zdania są rankingowane i szeregowane, a następnie wykorzystuje się je do utworzenia podsumowania. W przeciwieństwie do metod abstrakcyjnych, nie generują one nowego tekstu.

Scraping

Web Scraping to proces automatycznego pobierania i przetwarzania danych, które znajdują się na stronie internetowej. Stosuje się go w celu automatycznego wydobywania interesujących nas informacji.

Tokenizacja

Pierwszym krokiem w przetwarzaniu tekstu jest tokenizacja danych wejściowych. Otrzymany dokument dzielony jest na zdania, a one następnie rozdzielane na tzw. tokeny reprezentujące słowa, liczby i inne znaki tekstu. Aby analiza tych danych była miarodajna, zbiór należy znormalizować poprzez transformację do małych liter. Następnie odfiltrować należy elementy, które z reguły nie niosą żadnych istotnych informacji np. znaki interpunkcyjne.

Lematyzacja

Lematyzacja polega na wyodrębnieniu ze słowa w formie odmienionej jego bazowej formy lub słowa, od którego pochodzi forma odmieniona. Dzięki temu zbiór słów w dokumencie jest minimalizowany i pozwala na skuteczniejszą analizę podobieństwa pomiędzy poszczególnymi zdaniami.

Worek słów

Podzielony na tokeny dokument można przedstawić w postaci wektora liczb. Każdy element wektora reprezentuje częstotliwość występowania słowa w dokumencie.

TF-IDF

TF-IDF jest miarą, która ocenia znaczenie słowa w dokumencie w stosunku do zbioru dokumentów. Algorytm ma na celu wyważenie informacji o częstości wystąpienia elementu oraz jego znaczenia w kontekście całego zbioru dokumentów.

Wartość TF-IDF oblicza się na podstawie następującego wzoru:

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

Gdzie:

$$tf(t, d) = \frac{nt}{nd}, idf(t, D) = \log\left(\frac{ND}{(NDT)}\right),$$

t - słowo i liczba jego wystąpień nt

d – dokument i liczba występujących w nim słów nd

ND – liczba dokumentów

NDT - liczba dokumentów zawierających słowo t

Podobieństwo cosinusowe

Podobieństwo cosinusowe jest miarą używaną do określania podobieństwa między dwoma wektorami w przestrzeni wielowymiarowej. W kontekście przetwarzania języka naturalnego pozwala określić jak dwa teksty, przedstawione w postaci wektorów liczbowych, są do siebie podobne. Wzór na podobieństwo cosinusowe wygląda następująco:

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

PageRank

Algorytm PageRank z biblioteki NetworkX oparty został o algorytm użyty w wyszukiwarce Google do rankingowania stron internetowych. PageRank służy do obliczenia wag węzłów w grafie skierowanym na podstawie ilości krawędzi, które do niego dochodzą i wag węzłów od których pochodzą krawędzi.

Część praktyczna

Scrapping

Metoda służąca do przetwarzania danych tekstowych pobranych z witryny internetowej

```
def scrape_text_from_url(self) -> tuple[string, string, list[string]]:
    response = requests.get(self.url_path)
    if response.status_code == 200:
        soup = BeautifulSoup(response.content, 'html.parser')
        h1_title = soup.find('h1', class_='title').get_text() if soup.find('h1', class_='title') else 'No title found'
        p_lead = soup.find('p', class_='lead').get_text() if soup.find('p', class_='lead') else 'No lead found'
        div_content_parts = soup.find('div', class_='contentparts')
        p_content_parts = []
        if div_content_parts:
            for p in div_content_parts.find_all('p', class_='contentpart--text'):
                text = p.get_text()
                if '\xa0' not in text and 'ZOBACZ WIDEŃ' 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}")
```

Tokenizowanie i przygotowanie danych

Proces przygotowania danych obejmuje normalizację tekstu do małych liter, usunięcie znaków specjalnych nieniosących informacji oraz lematyzację.

```
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:
    morf = Morfeusz()
    text = text.lower()
    text = re.sub(r"\s+", " ", text).strip()
    text = re.sub(r"[\(\),]", "", text).strip()

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

Wektoryzacja

Do przetworzenia tekstu na wektory liczbowe udostępniono dwie metody.

```
def similarity_matrix_bow(self, sentences: list[list[str]]):
    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(sentences: list[list[str]], stop_words = get_polish_stopwords()) -> list[list[float]]:
    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
```

Ranking

Rankingowanie poszczególnych zdań wykonane jest za pomocą implementacji algorytmu PageRank z biblioteki NetworkX.

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

Generowanie podsumowania

Rankingowane zdania są szeregowane. Następnie wybierane są zdania o najwyższym wyniku. Liczba zdań podsumowania zależy od parametru początkowego.

```
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)]
```

Wyniki

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

Przykładowe podsumowanie artykułu dla techniki BOW:

- 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

Ten sam artykuł techniką 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

Podsumowanie

W raporcie przedstawiono projekt teoretyczny oraz implementację narzędzia do pobierania treści z artykułów w Internecie i tworzenia ich podsumowania opartego na najważniejszych informacjach. Wykorzystano techniki przetwarzania języka naturalnego takie jak normalizacja, lematyzacja. Przedstawiono techniki wektoryzacji poszczególnych tekstów i ich rankingowania w kontekście całości dokumentu.

Bibliografia

- [1] <https://towardsdatascience.com/understand-text-summarization-and-create-your-own-summarizer-in-python-b26a9f09fc70>
- [2] <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-przekształ%C5%82ca-s%C5%82owa-w-reprezentacje-liczbowe/>
- [3] <https://www.nltk.org>
- [4] <https://networkx.org>
- [5] <https://scikit-learn.org/stable>
- [6] <https://www.crummy.com/software/BeautifulSoup/bs4/doc>
- [7] Witold Kieraś, Marcin Woliński. Morfeusz 2 – analizator i generator fleksyjny dla języka polskiego. Język Polski, XCVII(1):75–83, 2017.
- [8] <https://github.com/bieli/stopwords/blob/master/polish.stopwords.txt>