

Narzędzie do tworzenia podsumowań tekstu

TK
SL
DL

1. Abstrakt

W niniejszej pracy przedstawiono narzędzie do automatycznego tworzenia podsumowań tekstu, opracowane przy użyciu języka Python oraz bibliotek ``nltk``, ``sumy``, ``glob`` i ``langdetect``. Narzędzie wykorzystuje algorytm TextRank do analizy i podsumowywania tekstu, co pozwala na wygenerowanie zwięzłego streszczenia wybranego dokumentu tekstowego. Użytkownik ma możliwość wyboru pliku tekstowego z bieżącego katalogu oraz określenia procentu tekstu, który ma zostać zachowany w podsumowaniu.

Główne zadania realizowane przez narzędzie obejmują:

Narzędzie identyfikuje język tekstu, zapewniając wsparcie dla popularnych języków takich jak angielski, polski, francuski, niemiecki, hiszpański i inne.

Tekst jest przetwarzany na pojedyncze słowa i zdania, co umożliwia dalszą analizę.

Algorytm TextRank analizuje tekst i generuje podsumowanie na podstawie zadanych parametrów.

Wynikowy tekst oraz statystyki dotyczące liczby słów w oryginalnym tekście i podsumowaniu są zapisywane w formie pliku HTML, który jest automatycznie otwierany w przeglądarce.

Narzędzie zostało zaprojektowane z myślą o prostocie użycia i efektywności, oferując użytkownikowi intuicyjny interfejs oraz szybkie wyniki. Dzięki zastosowaniu nowoczesnych metod analizy tekstu, narzędzie może być użyteczne w różnych dziedzinach, takich jak nauka, biznes, czy analiza danych. Zachęcamy do lektury, aby dowiedzieć się więcej o implementacji i potencjalnych zastosowaniach tego narzędzia.

2. Wstęp

2.1 Cel

Celem naszego projektu było stworzenie narzędzia do automatycznego tworzenia podsumowań tekstu w języku Python. Narzędzie to miało umożliwiać użytkownikom szybkie i efektywne generowanie streszczeń z dużych dokumentów tekstowych, zachowując kluczowe informacje i umożliwiając lepsze zrozumienie treści w krótszym czasie. Naszym zamiarem było również zapewnienie wsparcia dla wielu języków oraz umożliwienie użytkownikom dostosowania długości streszczenia do własnych potrzeb.

2.2 Zakres

Zakres projektu obejmował:

- Implementację detekcji języka tekstu przy użyciu biblioteki langdetect.
- Wykorzystanie bibliotek nltk oraz sumy do tokenizacji, parsowania oraz generowania streszczeń tekstu.
- Obsługę wielu języków.
- Umożliwienie użytkownikom wyboru poziomu streszczenia poprzez określenie procentowego poziomu zawartości tekstu do zachowania.
- Analizę statystyk streszczenia, takich jak liczba słów w tekście oryginalnym i streszczeniu.
- Zapisanie i wyświetlenie wyników w formacie HTML, co umożliwia łatwe przeglądanie i analizę wyników.

2.3 Metodyka

Do realizacji naszego projektu zastosowaliśmy następującą metodykę i narzędzia:

Analiza Wymagań:

- Zidentyfikowanie potrzeb użytkowników końcowych oraz specyfikacji funkcjonalnych narzędzia.

Wykorzystane Narzędzia:

- Python: Język programowania użyty do implementacji całego narzędzia.
- nltk: Biblioteka do przetwarzania języka naturalnego, użyta do tokenizacji i przetwarzania tekstu.
- sumy: Biblioteka do generowania podsumowań tekstu, w szczególności algorytm TextRank.
- langdetect: Biblioteka do detekcji języka tekstu.
- webbrowser oraz os: Moduły użyte do zapisywania i wyświetlania wyników w formacie HTML.
- glob: Moduł używany do znajdowania plików tekstowych w bieżącym katalogu.

Implementacja:

- Zaimplementowanie funkcji do detekcji języka tekstu, tokenizacji oraz parsowania tekstu.
- Integracja algorytmu TextRank do generowania podsumowań.
- Stworzenie mechanizmu pozwalającego użytkownikom na wybór poziomu streszczenia oraz analizę statystyk streszczenia.
- Implementacja funkcji do zapisywania wyników w formacie HTML i automatycznego otwierania ich w przeglądarce.

Testowanie i Walidacja:

- Przeprowadzenie testów na różnorodnych dokumentach tekstowych w różnych językach w celu zapewnienia poprawności działania narzędzia.
- Walidacja wyników pod kątem zgodności z oczekiwaniami użytkowników oraz jakości generowanych streszczeń.

Dokumentacja:

- Stworzenie dokumentacji opisującej sposób użycia narzędzia oraz jego funkcjonalności, umożliwiającej użytkownikom łatwe rozpoczęcie pracy z narzędziem.

3. Część teoretyczna

Przetwarzanie Języka Naturalnego (NLP) to dziedzina informatyki zajmująca się interakcjami między komputerami a językiem ludzkim. Celem NLP jest umożliwienie komputerom zrozumienia, interpretacji i generowania języka ludzkiego w sposób wartościowy. W naszym projekcie wykorzystaliśmy różne techniki NLP, takie jak tokenizacja, wykrywanie języka i generowanie podsumowań.

Tokenizacja jest procesem dzielenia tekstu na mniejsze jednostki zwane tokenami, które mogą być słowami, zdaniami lub innymi jednostkami lingwistycznymi. W naszym narzędziu użyliśmy tokenizacji do analizy tekstu i jego przygotowania do dalszego przetwarzania. Wykorzystaliśmy bibliotekę nltk, która oferuje różnorodne narzędzia do tokenizacji w różnych językach.

Detekcja języka jest kluczowym elementem naszego narzędzia, ponieważ umożliwia dostosowanie procesów przetwarzania tekstu do specyfiki językowej. Użyliśmy biblioteki langdetect, która bazuje na algorytmie n-gramów, aby automatycznie identyfikować język wejściowego tekstu. Detekcja języka pozwala na zastosowanie odpowiednich narzędzi tokenizacji i analizy tekstu zgodnie z jego językiem.

Analiza statystyk tekstu, takich jak liczba słów w oryginalnym tekście i w streszczeniu, pozwala na ocenę skuteczności i jakości generowanych podsumowań. W naszym narzędziu zaimplementowaliśmy funkcje do liczenia słów i porównywania tych wartości między tekstem wejściowym a jego podsumowaniem. Dzięki temu użytkownicy mogą łatwo ocenić, jak duża część oryginalnego tekstu została zachowana w streszczeniu.

Algorytm TextRank, zaproponowany przez Rada Mihalcea i Paula Tarau w 2004 roku, jest metodą analizy grafowej, która została pierwotnie zaprojektowana do automatycznego streszczania tekstu i ekstrakcji słów kluczowych. TextRank jest inspirowany algorytmem PageRank, który Google wykorzystuje do rankingowania stron internetowych w wynikach wyszukiwania. TextRank opiera się na założeniu, że najważniejsze elementy (zdania, frazy) w tekście mogą być zidentyfikowane na podstawie ich relacji z innymi elementami. Relacje te są reprezentowane jako graf, gdzie węzły to elementy tekstu (np. zdania), a krawędzie reprezentują podobieństwo między nimi. W kontekście streszczania tekstu, poszczególne zdania w dokumencie są traktowane jako węzły w grafie. Krawędzie między węzłami są tworzone na podstawie podobieństwa zdań, które można mierzyć na różne sposoby, np. poprzez współdzielone słowa. Waga krawędzi odzwierciedla stopień podobieństwa między dwoma zdaniami.

TextRank wykorzystuje iteracyjny proces podobny do PageRank do obliczenia ważności każdego węzła. Proces ten można opisać następującymi krokami:

Inicjalizacja: Przypisanie początkowej wartości rangi każdemu węzłowi. Zazwyczaj wszystkie węzły zaczynają z taką samą wartością.

Aktualizacja: Wartość rangi każdego węzła jest aktualizowana na podstawie wartości rang sąsiednich węzłów, uwzględniając wagi krawędzi. Proces ten powtarza się iteracyjnie, aż do osiągnięcia zbieżności.

Po zakończeniu iteracyjnego procesu, węzły (zdania) są sortowane według ich wartości rang. Najwyżej oceniane zdania są wybierane do tworzenia streszczenia. Liczba wybranych zdań może być dostosowana w zależności od pożądanego poziomu streszczenia.

TextRank nie wymaga danych uczących ani wstępnej wiedzy o języku, co czyni go elastycznym i łatwym do wdrożenia.

Jakość streszczeń: Dzięki uwzględnieniu relacji między zdaniami, TextRank generuje streszczenia, które zachowują kontekst i spójność.

Wsparcie wielojęzyczne: Algorytm może być stosowany do tekstów w różnych językach, pod warunkiem odpowiedniego przetwarzania wstępnego.

W naszym projekcie algorytm TextRank został zaimplementowany przy użyciu biblioteki sumy.

4. Część praktyczna

Rozwiązanie problemu automatycznego tworzenia podsumowań tekstu w naszym projekcie opiera się na zastosowaniu algorytmu TextRank oraz kilku kluczowych technik przetwarzania języka naturalnego. Nasze narzędzie zostało zaprojektowane tak, aby było uniwersalne i łatwe w użyciu, a także aby wspierało wiele języków. W tej części omówimy szczegółowo proces implementacji, prezentację wyników oraz dyskusję na temat skuteczności i możliwości rozwoju narzędzia.

Implementacja rozpoczęła się od analizy wymagań użytkowników oraz specyfikacji funkcjonalnych narzędzia. Kluczowym elementem było wsparcie dla wielojęzyczności, co osiągnęliśmy dzięki wykorzystaniu biblioteki langdetect do automatycznego wykrywania języka tekstu wejściowego. Detekcja języka była niezbędna, aby móc odpowiednio dostosować procesy tokenizacji i analizy tekstu do specyfiki każdego języka. Po wykryciu języka, tekst był tokenizowany na zdania przy użyciu tokenizerów z biblioteki nltk, co umożliwiało dalsze przetwarzanie.

Kolejnym krokiem było wykorzystanie algorytmu TextRank do generowania podsumowań tekstu. Algorytm ten, bazujący na grafowej analizie relacji między zdaniami, pozwala na identyfikację najważniejszych zdań w dokumencie. Implementacja TextRank przy użyciu biblioteki sumy umożliwiła nam łatwe obliczenie wartości rang dla poszczególnych zdań i wybór tych najbardziej znaczących. Użytkownik miał możliwość określenia procentowego poziomu streszczenia, co pozwalało na dostosowanie długości podsumowania do własnych potrzeb. Wyniki generowanych podsumowań były prezentowane użytkownikom w formacie HTML, co ułatwiało ich przeglądanie i analizę. Zaimplementowaliśmy mechanizm do zapisywania wyników w plikach HTML oraz automatycznego otwierania ich w przeglądarce internetowej. Dzięki temu użytkownicy mogli w intuicyjny sposób porównać oryginalny tekst z jego streszczeniem oraz zobaczyć statystyki podsumowania, takie jak liczba słów w tekście oryginalnym i streszczeniu.

Podczas testowania narzędzia na różnorodnych dokumentach tekstowych w różnych językach zaobserwowaliśmy wysoką skuteczność algorytmu TextRank w generowaniu spójnych i sensownych podsumowań. Streszczenia zachowywały kluczowe informacje i były zrozumiałe, co potwierdzało trafność wyboru TextRank jako podstawy naszego narzędzia. Jednakże, zauważyliśmy również pewne ograniczenia. Na przykład, algorytm TextRank może mieć trudności z przetwarzaniem bardzo długich dokumentów lub tekstów o bardzo złożonej strukturze, co może wpływać na jakość generowanych podsumowań. Podsumowując, nasze narzędzie do automatycznego tworzenia podsumowań tekstu spełniło postawione przed nim cele. Umożliwia szybkie i efektywne generowanie streszczeń dla dokumentów w wielu językach, zachowując przy tym wysoką jakość i spójność. Ostateczne wyniki wskazują na dużą użyteczność narzędzia, jednak przyszłe prace mogą skupić się na dalszej optymalizacji algorytmu oraz rozszerzeniu jego możliwości, na przykład poprzez integrację z innymi technikami NLP lub uczeniem maszynowym, aby jeszcze lepiej radzić sobie z bardziej złożonymi tekstami.

Omówienie działania skryptu

1. Ustawianie deterministyczności dla wykrywania języka i pobieranie zasobu dla tokenizacji.

```
DetectorFactory.seed = 0
nltk.download('punkt')
```

2. Ustawianie zestawu obsługiwanych języków.

```
SUPPORTED_LANGUAGES = {'en', 'pl', 'fr', 'de', 'es', 'it', 'ru', 'zh-cn', 'ja', 'pt', 'nl', 'ar'}
```

3. Ta funkcja wykrywa język danego tekstu i sprawdza, czy jest on obsługiwany.

```
def detect_language(text):
    try:
        detected_lang = detect(text)
        if detected_lang in SUPPORTED_LANGUAGES:
            return detected_lang
        else:
            print(f"Detected language '{detected_lang}' is not supported. Defaulting to English.")
            return 'en'
    except Exception as e:
        print(f"An error occurred during language detection: {e}")
        return 'en'
```

4. Tokenizacja tekstu na pojedyncze słowa z wykorzystaniem biblioteki nltk.

```
def tokenize_text(text):
    words = word_tokenize(text.lower())
    return words
```


5. Ta funkcja generuje podsumowanie w formie pliku HTML i otwiera go w przeglądarce. Zawiera również statystyki dotyczące liczby słów w oryginalnym tekście i podsumowaniu.

```
def save_and_display_summary(original_text, summary_text, original_word_count, summary_word_count, file_name):
    html_content = f"""
    <html>
    <head>
        <title>Text Summary</title>
        <style>
            body {{ font-family: Arial, sans-serif; line-height: 1.6; margin: 20px; text-align: center; }}
            .box {{ border: 1px solid #ccc; padding: 20px; border-radius: 10px; margin-bottom: 20px; display: inline-block; text-align: left; width: 60%; }}
            .centered-box p {{ text-align: center; }}
        </style>
    </head>
    <body>
        <h2>Original Text</h2>
        <div class="box">
            <p>{original_text.replace('\n', '<br>')}</p>
        </div>
        <h2>Summary</h2>
        <div class="box">
            <p>{summary_text.replace('\n', '<br>')}</p>
        </div>
        <h2>Summary Statistics</h2>
        <div class="box centered-box">
            <p>Original word count: {original_word_count}</p>
            <p>Summary word count: {summary_word_count}</p>
        </div>
    </body>
    </html>
    """
    file_path = file_name.replace('.txt', '.html')
    with open(file_path, 'w', encoding='utf-8') as file:
        file.write(html_content)

    webbrowser.open('file://' + os.path.realpath(file_path))
```

6. Funkcja `summarize_with_text_rank` rozpoczyna się od otwarcia pliku tekstowego, odczytania jego zawartości i detekcji języka tekstu. Następnie tekst jest parsowany i tokenizowany, a algorytm TextRank Summarizer jest inicjalizowany. Liczba zdań w tekście jest obliczana, a na podstawie procentowego udziału tekstu w podsumowaniu obliczana jest liczba zdań, które będą stanowiły podsumowanie. Algorytm generuje podsumowanie, które następnie jest przekształcane w ciąg znaków. Tekst oryginalny i jego podsumowanie są tokenizowane, a następnie obliczane są statystyki słów. Ostatecznie wywoływana jest funkcja `save_and_display_summary`, która zapisuje wyniki w formie pliku HTML i otwiera go w przeglądarce. W przypadku wystąpienia błędu, funkcja wyświetla odpowiedni komunikat.

```
def summarize_with_text_rank(file_path, summary_percentage):
    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            text = file.read()

        language = detect_language(text)
        print(f"Detected language: {language}")

        parser = PlaintextParser.from_string(text, Tokenizer(language))
        summarizer = TextRankSummarizer()

        total_sentences = len(list(parser.document.sentences))
        summary_sentence_count = max(1, int(total_sentences * (summary_percentage / 100)))

        summary = summarizer(parser.document, summary_sentence_count)

        summary_text = " ".join([str(sentence) for sentence in summary])
        print("\nSummary:", summary_text)

        original_words = tokenize_text(text)
        summary_words = tokenize_text(summary_text)

        original_word_count = len(original_words)
        summary_word_count = len(summary_words)

        print(f"\nOriginal word count: {original_word_count}")
        print(f"Summary word count: {summary_word_count}")

        save_and_display_summary(text, summary_text, original_word_count, summary_word_count, file_path)
    except Exception as e:
        print(f"An error occurred: {e}")
```

5. Podsumowanie

Celem naszego projektu było stworzenie narzędzia do automatycznego tworzenia podsumowań tekstu, które wspierałoby wiele języków i było łatwe w użyciu. Cele te zostały w pełni osiągnięte. Nasze narzędzie, bazujące na algorytmie TextRank, efektywnie generuje spójne i sensowne podsumowania, co potwierdziły testy przeprowadzone na różnorodnych dokumentach tekstowych. Wykorzystanie bibliotek `nltk`, `sumy` i `langdetect` umożliwiło nam wsparcie dla różnych języków oraz przeprowadzenie analizy i przetwarzania tekstu na wysokim poziomie.

Implementacja wsparcia wielojęzycznego oraz interfejsu użytkownika, który prezentuje wyniki w przystępnej formie HTML, pozwoliła na stworzenie narzędzia, które jest zarówno funkcjonalne, jak i intuicyjne w obsłudze. Wyniki naszych testów wykazały, że narzędzie działa skutecznie i jest w stanie generować podsumowania, które zachowują kluczowe informacje zawarte w oryginalnych dokumentach.

Jednakże, podczas realizacji projektu natrafiliśmy na pewne wyzwania. Algorytm TextRank ma swoje ograniczenia, szczególnie w przypadku bardzo długich dokumentów lub tekstów o złożonej strukturze, co czasami wpływało na jakość generowanych podsumowań. Mimo to, narzędzie okazało się być użyteczne i efektywne dla większości testowanych przypadków.

W przyszłości możliwe jest dalsze rozwijanie narzędzia, np. poprzez integrację z innymi technikami przetwarzania języka naturalnego lub uczeniem maszynowym, co mogłoby poprawić jego skuteczność w bardziej złożonych przypadkach. Subiektywnie oceniamy, że projekt był wartościowym doświadczeniem, które pozwoliło nam zgłębić zarówno teoretyczne, jak i praktyczne aspekty NLP oraz rozwinąć umiejętności w zakresie programowania i analizy danych.

6. Bibliografia

- Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing Order into Texts. Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing
- Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media.
- Python Software Foundation. (n.d.). NLTK 3.6.5 documentation.
- Python Software Foundation. (n.d.). Langdetect 1.0.9 documentation.
- Python Software Foundation. (n.d.). Sumy 0.8.1 documentation.
- Brin, S., & Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems, 30(1-7), 107-117.