Text summarizer

Narzędzie do tworzenia podsumowań tekstu

Autorzy:
- Natalia Bieniek
- Amelia Kwiecińska
- Adam Kwoka

Text summarizer, czyli narzędzie do tworzenia podsumowań tekstu

Naszym zadaniem było stworzenie narzędzia do tworzenia podsumowań tekstu. Jest to problem polegający na zredukowaniu ilości tekstu bez zmieniania znaczenia całości. Istnieje wiele różnych technik, za pomocą których można uzyskać taki efekt. Możemy wyróżnić dwie kategorie takich metod:

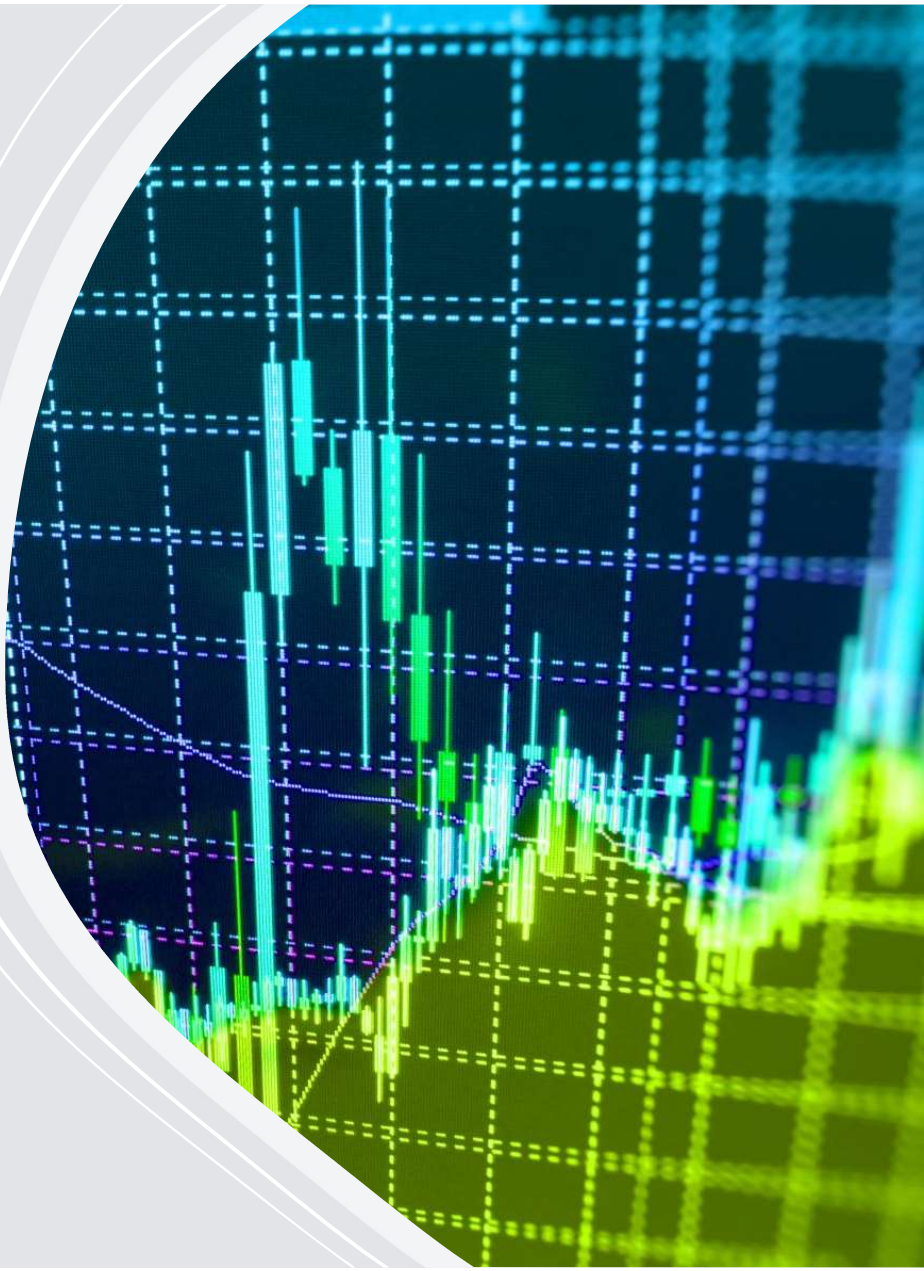
- Ekstrakcyjny – polega na wybraniu najważniejszych zdań z tekstu i utworzenie pod zbioru składającego się z części całości jednocześnie zachowując przekaz płynący z całości
- Abstrakcyjny – wykorzystuje zaawansowane przetwarzanie języka naturalnego i tworzy podsumowanie tekstu na podstawie całości. Jest to trudniejsza technika, ponieważ wymaga wielu zmiennych oraz danych na samym początku tworzenia algorytmu.

Ekstrakcyjne podejście

W rozwiązaniu tego problemu wykorzystaliśmy podejścia ekstrakcyjnego, polegającego na wyciąganiu najważniejszych zdań z całości tekstu i tworzeniu dzięki nim podsumowania zachowującego sens całości tekstu.

W tym celu posłużyliśmy się następującymi metodami:

- Metoda częstotliwości
- Wykorzystanie pakietu Sumy
    - Sumaryzacja z wykorzystaniem TextRank z pakietu Sumy
    - Sumaryzacja z wykorzystaniem LunhAlgorithm z pakietu Sumy



```python
# Tworzenie summarizer z wykorzystaniem metody częstotliwości
def textSummarizerWithFrequencyMethod(text):
    # Ustawianie stopwords
    settedStopwords = set(stopwords.words("english"))
    # Tokenizowanie słów
    words = word_tokenize(text)

    # Tworzenie tablicy częstotliwości występowania słów
    frequncyTable = {}
    for word in words:
        word = word.lower()
        if word in settedStopwords:
            continue
        if word in frequncyTable:
            frequncyTable[word] += 1
        else:
            frequncyTable[word] = 1

    # Tokenizowanie sentencji
    sentences = sent_tokenize(text)

    # Tworzenie tablicy częstotliwości dla sentencji z wykorzystaniem częstotliwości słów
    sentenceValue = {}
    for sentence in sentences:
        for word, frequnece in frequncyTable.items():
            if word in sentence.lower():
                if sentence in sentenceValue:
                    sentenceValue[sentence] += frequnece
                else:
                    sentenceValue[sentence] = frequnece

    # Tworzenie średniej na podstawie której będziemy tworzyć podsumowanie
    sumSentencesValues = 0
    for valueOfSentence in sentenceValue:
        sumSentencesValues += sentenceValue[valueOfSentence]
    average = int(sumSentencesValues / len(sentenceValue))

    # Tworzenie podsumowania tekstu. Możemy dowolnie ustawiać wartość dokładności z jaką ma być uwtorzone podsumowanie
    summaryText = ''
    for sentence in sentences:
        if sentence in sentenceValue and sentenceValue[sentence] > (1.19 * average):
            summaryText += " " + sentence

    return summaryText
```

Metoda częstotliwości

Naszym pierwszym rozwiązaniem jest wykorzystanie metody częstotliwości. Polega ona na tym, że znajdujemy częstotliwość wszystkich słów w naszych danych tekstowych i przechowujemy dane tekstowe oraz ich częstotliwości w słowniku. Następnie tokenizujemy nasze dane tekstowe. Zdanie, które zawierają więcej słów o wysokiej częstotliwości, zostają zachowane w końcowym podusmowaniu.

```python
# Tworzenie summarizer z wykorzystaniem Sumy text_rank
def textSummarizerWithSumyTextRank(text):
    # Parsowanie tekstu
    parser = PlaintextParser.from_string(text, Tokenizer("english"))

    # Tworzenie podsumowania za pomocą sumy TextRank
    summarizer = TextRankSummarizer()
    summaryList = summarizer(parser.document, 2)

    summary = ""
    for sentence in summaryList:
        summary += " " + str(sentence)
    return summary
```

Sumaryzacja z wykorzystaniem TextRank z pakietu Sumy

Sumy jest to algorytm uczenia maszynowego oparty na TextRank. Wykorzystanie tego algorytmu jest bardzo proste i pozwala w bardzo szybki sposób utworzyć podsumowanie tekstu, używając ekstrakcyjnego podjeścia.

```python
# Tworzenie summarizer z wykorzystaniem Sumy lunh_algorithm
def textSummarizerWithLunhAlgorithm(text):
    # Parsowanie tekstu
    parser = PlaintextParser.from_string(text, Tokenizer("english"))

    # Tworzenie podsumowania za pomocą sumy TextRank
    summarizer_luhn = LuhnSummarizer()
    summaryList = summarizer_luhn(parser.document, 2)

    dp = []
    for i in summaryList:
        lp = str(i)
    dp.append(lp)

    summary = ' '.join(dp)
    return summary
```

Sumaryzacja z wykorzystaniem LunhAlgorithm z pakietu Sumy

Naszym trzecim rozwiązaniem problemu było wykorzystanie sumaryzacji z wykorzystaniem LunhAlgorithm. Podejście to opiera się na metodzie częstotliwości, czyli jest to rozwiązanie bliskie pierwszemu rozwiązaniu.

Obserwacje

W trakcie rozwiązywania problemu narzędzia do tworzenia podsumowania tekstu musieliśmy rozważyć następujące ścieżki. Jedną z nich było podejście ekstrakcyjne, które polega na wybieraniu najważniejszych zdań z całości i tworzenie na tej podstawie streszczenia zachowującego sens całości tekstu. Drugim było podejście abstrakcyjne, opierające się na wielu danych i zmiennych, dzięki któremu tworzone jest streszczenie wynikające z sensu całości tekstu nie bazując na gotowych zdaniach pochodzących z oryginalnego tekstu.

Zdecydowaliśmy pójść drogą ekstrakcyjną, ponieważ jest ona prostsza, lecz niemniej skuteczna. Po uzyskaniu danych wyjściowych sens tekstu jest zachowany oraz przedstawiony w zwięzłym formacie.

Bibliografia

- https://towardsdatascience.com/text-summarization-with-nlp-textrank-vs-seq2seq-vs-bart-474943efeb09
- https://github.com/edubey/text-summarizer/blob/master/text-summarizer.py
- https://www.machinelearningplus.com/nlp/text-summarization-approaches-nlp-example/
- https://www.analyticssteps.com/blogs/what-text-summarization-nlp