Raport

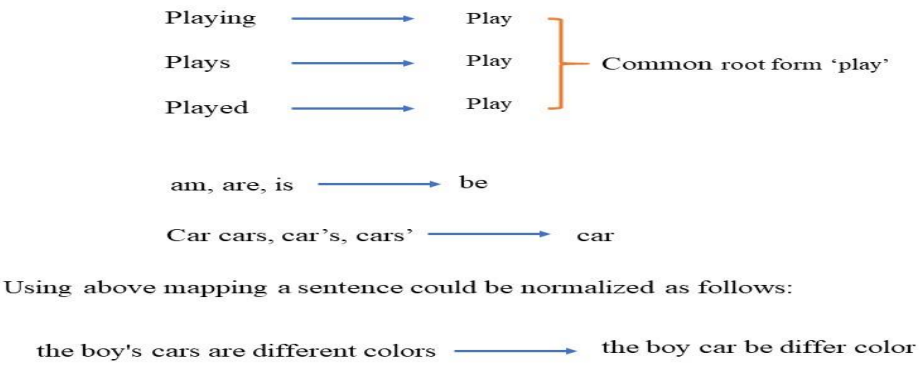
Wykonywanie procesu stemming-u na wybranych tekstach w języku angielskim

Autor: Damian Schröder

Data przygotowania: 17.11.2021r.

Spis treści

Stemming pochodzi od korzenia słowa (ang. *'stem')*. Jest to technika normalizacji tekstu (lub słowa), która polega na otrzymaniu różnych morfologicznych wariacji danego słowa posiadając jego *'stem'*, np. wyraz 'wait' posiada takie samo znaczenie jak 'waiting', 'waited', 'waits'. Ta metoda przetwarzania wstępnego pozwala, m.in. na łatwiejsze wyszukiwanie słów w wyszukiwarkach.



DataCamp

Narzędzia

Porter Stemmer: `from nltk.stem import PorterStemmer`

Jest to najstarsze narzędzie używane do stemmingu stworzone przez Martina Portera w 1980. Znane jest ze swojej prostoty i szybkości, a jego rezeultatem jest najczęściej skrócone słowo, zachowujące znaczenie jego korzenia.

Lancaster Stemmer: `from nltk.stem import LancasterStemmer`

Proste narzędzie którego wyniki często są zbytnio 'przycięte', przez co zazwyczaj przestają być poprawne lingwistycznie lub tracą znaczenie.

Snowball Stemmer: `from nltk.stem.snowball import SnowballStemmer`

Stworzone również przez Martina Portera, lecz algorytm użyty tutaj jest bardziej dokładny, oferuje nieduże ulepszenie pod względem logiki, jak i szybkości. Potocznie zwany „Porter2 Stemmer" lub „English Stemmer".

Krovetz Stemmer: `from krovetzstemmer import Stemmer`

Został zaproponowany przez Roberta Krovetz'a w 1993. Zasady działania tego alogrytmu polegają na sprowadzeniu z formy mnogiej, do pojedynczej oraz z czasu przeszłego na teraźniejszy, a następnie usunięcie końcówki 'ing'.

Regexp Stemmer: `from nltk.stem import RegexpStemmer`

Używa wyrażeń regularnych do identyfikowania poszczególnych wzorców. Każda część wyrazu, która będzie się wpasowywała w to wyrażenie, będzie usuwana.

Brak

Lovins Stemmer, Dawson Stemmer, Xerox Stemmer

Dodatkowe paczki:

pandas: `import pandas as pd`
word_tokenize: `from nltk.tokenize import word_tokenize`

Używane dane

SMSSpamCollection – PUBLICZNA KOLEKCJA SMS SKŁADAJĄCA SIĘ Z DWÓCH KOLUMN, GDZIE PIERWSZA („HAM"/"SPAM") OZNACZA CZY JEST TO SPAM, BĄDŹ NIE, NATOMIAST DRUGA ZAWIERA SAMĄ TREŚĆ WIADOMOŚCI

HTTPS://ARCHIVE-BETA.ICS.UCI.EDU/ML/DATASETS/SMS+SPAM+COLLECTION

Została ona dostosowana za pomocą pakietu PANDAS. Szczegóły poniżej.

Eksperymenty

Do pobrania danych z pliku zostało użyta funkcja READ_CSV *z pakietu* PANDAS

```
df = pd.read_csv("SMSSpamCollection", delimiter='\t', names=['ID',
'MESSAGE'])
```

Następnie została odseparowana sama kolumna, w której znajdowały się same treści wiadomości, potem poszczególne linie, a z nich wyrazy

```
tresc = df['MESSAGE']
all_words = []
for t in tresc:
    words = word_tokenize(t)
    for w in words:
        all_words.append(w)
```

Kolejnym krokiem było wyłączenie powtarzających się wyrazów za pomocą NLTK.FREQDIST, najczęściej powtarzających się wyrazów, które nie oferują zbyt wiele informacji z użyciem NLTK.CORPUS.STOPWORDS('ENGLISH') oraz upewnienie się, że wyrazy zawierają wyłącznie litery i są długości przynajmniej czterech liter

```
all_words = nltk.FreqDist(all_words)
all_words = [x for x in all_words if x not in
set(nltk.corpus.stopwords.words('english')) and x.isalpha() and len(x) > 3]
```

Ostatnim krokiem była stematyzacja naszego zbioru za pomocą wyżej wymienionych narzędzi

```
print('Liczba słów: {}'.format(len(all_words)))
print("{0:20}{1:20}{2:20}{3:20}{4:20}{5:20}".format("Word","Porter
Stemmer","Lancaster Stemmer", "Snowball Stemmer", "Krovetz Stemmer",
"Regexp Stemmer"))
for w in all_words:

print("{0:20}{1:20}{2:20}{3:20}{4:20}{5:20}".format(w,porter.stem(w),lancas
ter.stem(w),snowballeng.stem(w),krovetz.stem(w),regexp.stem(w)))
```

A tak prezentują się wybrane wyniki na naszych danych

```
Liczba słów: 7832
Word                Porter Stemmer      Lancaster Stemmer   Snowball Stemmer    Krovetz Stemmer     Regexp Stemmer
call                call                cal                 call                call                call
know                know                know                know                know                know
like                like                lik                 like                like                lik
come                come                com                 come                come                com
time                time                tim                 time                time                tim
want                want                want                want                want                want
Call                call                cal                 call                call                Call
home                home                hom                 home                home                hom
going               go                  going               go                  go                  go
send                send                send                send                send                send
good                good                good                good                good                good
need                need                nee                 need                need                need
love                love                lov                 love                love                lov
back                back                back                back                back                back
still               still               stil                still               still               still
text                text                text                text                text                text
later               later               lat                 later               later               later
```

Analiza

W celu przeprowadzenia analizy pogrupowałem użyte stematyzatory w pary i następnie porównałem je ze sobą.

W tym celu utworzyłem listy zawierające dane słowa po procesie stematyzacji

```
porownarka = []
for w in all_words:
    temp = [porter.stem(w),lancaster.stem(w),snowballeng.stem(w),krovetz.stem(w),regexp.stem(w)]
    print("{0:20}{1:20}{2:20}{3:20}{4:20}{5:20}".format(w,porter.stem(w),lancaster.stem(w),snowballeng.stem(w),krovetz.stem(w),regexp.stem(w)))
    porownarka.append(temp)
```

Dalszym moim posunięciem było stworzenie dwuwymiarowej tablicy (listy), w celu umieszczenia w niej liczby słów, które dane stematyzatory zmieniły (bądź zostawiły) w taki sam sposób

```
pary = []
for x in range(5):
    col = []
    for y in range(5):
        col.append(0)
    pary.append(col)

for word in porownarka:
    for w in range(0,5):
        if word[0] == word[w]:
            pary[0][w]+=1
        if word[1] == word[w]:
            pary[1][w]+=1
        if word[2] == word[w]:
            pary[2][w]+=1
        if word[3] == word[w]:
            pary[3][w]+=1
        if word[4] == word[w]:
            pary[4][w]+=1
```

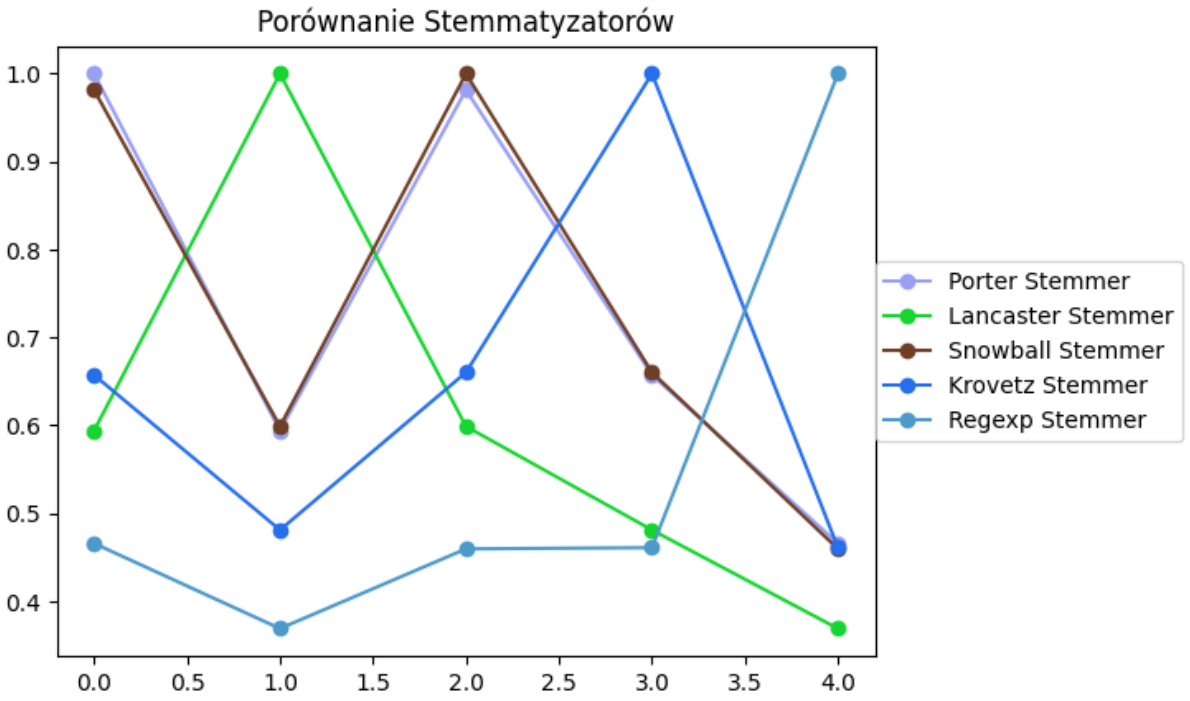
Podzielenie wyników na liczbę wszystkich badanych słów, by otrzymać skalę procentową i wyświetlenie rezultatów

```
for x in range(5):
    for y in range(5):
        pary[x][y] /= len(all_words)


print("")
print("{:8}".format("Stemmatyzer Porter Lancaster Snowball Krovetz Re-
gexp"))
nazwy = ["Porter","Lancaster", "Snowball", "Krovetz", "Regexp"]

i = 0
for row in pary:
    print("{:8}".format(nazwy[i]), end=" ")
    for col in row:
        print("{:8.3f}".format(col), end=" ")
    print("")
    i+=1
```

Stemmatyzer	Porter	Lancaster	Snowball	Krovetz	Regexp
Porter	1.000	0.594	0.981	0.657	0.466
Lancaster	0.594	1.000	0.599	0.481	0.369
Snowball	0.981	0.599	1.000	0.660	0.460
Krovetz	0.657	0.481	0.660	1.000	0.461
Regexp	0.466	0.369	0.460	0.461	1.000

Na podstawie zebranych danych można również stworzyć wykres za pomocą PYPLOT z PAKIETU MATPLOTLIB.PYPLOT, pomocne w tym będzie również NUMPY

```
fig = plt.figure(1)
nazwy = ["Porter Stemmer","Lancaster Stemmer", "Snowball Stemmer", "Krovetz
Stemmer", "Regexp Stemmer"]
plt.title("Porównanie Stemmatyzatorów")
x = np.array(range(0,5))
for i, array in enumerate(pary):
    plt.plot(x, array, color = np.random.rand(3, ), marker = "o", label =
nazwy[i])

plt.legend(loc = "center left", bbox_to_anchor=(1, 0.5), borderaxespad=0.)
fig.savefig('wykres', bbox_inches='tight')
plt.show()
```



Wyniki są w większości takie, jakich powinniśmy się spodziewać. Lancaster Stemmer często działa zbyt nadgorliwie, przez co dane słowo traci swoje oryginalne znaczenie lub całkowicie. Regexp Stemmer działa według podanych mu instrukcji. Stemmery Porter oraz Snowball mają wysoki współczynnik podobieństwa rezultatów (~98%), co nie jest zaskoczeniem, gdyż są dziełem jednego twórcy. Porter, Snowball i Krovetz Stemmer to grupa, która wykazuje wyższe od reszty testowanych stemmerów podobieństwa otrzymanych rezultatów (~65%).