SW04 - Gruppe 1

NLP Basics 2

M. Nebroj, S. Hauri, S. Ineichen, R. Schwarzentruber

2019-10-12

Testatübung SW04

Aufgabe 1 - N-Grams

Identifiziert die Bi- und Trigrams aus `shakespeare-macbeth.txt`. Dieses Buch ist Teil des Gutenberg Corpus. Der Datensatz muss zunächst bereinigt werden (Stop Words, Satzzeichen, etc.)

Code - Vorbereitung

```
import nltk, re, string, collections
from nltk.util import ngrams
from nltk.corpus import stopwords

# Download Stopwords
nltk.download('stopwords')

# Download MacBeth from Gutenberg
nltk.download('gutenberg')
nltk.corpus.gutenberg.fileids()
['shakespeare-macbeth.txt']

macbeth = nltk.corpus.gutenberg.words('shakespeare-macbeth.txt')

## Exercise 1

# Satzzeichen entfernen
text_ohne_satzzeichen = "[" + re.sub("\.","",string.punctuation) + "]"
macbeth_ohne_satzzeichen = []

for word in macbeth:
    word = re.sub(text_ohne_satzzeichen, "", word)
    if word != "":
        macbeth_ohne_satzzeichen.append(word)

# Stopwörter entfernen
stop_words = set(stopwords.words('english'))

token_clean = [w for w in macbeth_ohne_satzzeichen if not w in stop_words]
```

Code - Bigrams

```
# Bigrams definieren
listBigrams = nltk.bigrams(token_clean)

freq_bi = nltk.FreqDist(listBigrams)
```

```
fdist = nltk.FreqDist(freq_bi)
print("\n\nBigrams: \n")
for k,v in fdist.most_common():
    if v > 8:
        print (k,v)
```

Output - Bigrams

```
Bigrams:

('I', 'haue') 41
('Exeunt', 'Scena') 15
('Enter', 'Macbeth') 15
('Thane', 'Cawdor') 13
('Macb', 'I') 13
('I', 'would') 11
('yet', 'I') 10
('st', 'thou') 9
('And', 'yet') 9
('I', 'see') 9
('Knock', 'Knock') 9
```

Code - Trigrams

Trigrams definieren

```
listTrigrams = nltk.trigrams(token_clean)

freq_tri = nltk.FreqDist(listTrigrams)

fdist = nltk.FreqDist(freq_tri)
print("\n\nTrigrams: \n")
for k,v in fdist.most_common():
    if v > 2:
        print (k,v)
```

Output - Trigrams

```
Trigrams:

('Knock', 'Knock', 'Knock') 5
('I', 'good', 'Lord') 5
('Enter', 'three', 'Witches') 4
('Exeunt', 'Scena', 'Secunda') 4
('I', 'haue', 'seene') 4
('I', 'see', 'thee') 4
('Enter', 'Macbeth', 'Macb') 4
('I', 'haue', 'done') 4
('good', 'Lord', 'Macb') 4
('three', 'Witches', '1') 3
('Exeunt', 'Scena', 'Tertia') 3
('Thunder', 'Enter', 'three') 3
('All', 'haile', 'Macbeth') 3
('Exeunt', 'Scena', 'Quarta') 3
('But', 'I', 'haue') 3
('Scena', 'Prima', 'Enter') 3
('Scena', 'Secunda', 'Enter') 3
```

```
('All', 'Double', 'double') 3
('trouble', 'Fire', 'burne') 3
('Fire', 'burne', 'Cauldron') 3
('burne', 'Cauldron', 'bubble') 3
('Enter', 'Malcolme', 'Seyward') 3
```

Aufgabe 2 - TF-IDF Wert der Bigrams

Berechnet den TF-IDF Wert der Bigrams aus den Dokumenten IMDB_1.txt, IMDB_2.txt, IMDB_3.txt

Code

```
## Exercise 2

import numpy as np
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

imdb_1 = open('IMDB_1.txt', 'r').readline()
imdb_2  = open('IMDB_2.txt', 'r').readline()
imdb_3  = open('IMDB_2.txt', 'r').readline()

documents = [
    imdb_1,
    imdb_2,
    imdb_3
]

document_names = ['IMDB {:d}'.format(i+1) for i in range(len(documents))]

def get_tfidf(docs, ngram_range=(1,2), index=None):
    vect = TfidfVectorizer(stop_words='english', ngram_range=ngram_range)
    tfidf = vect.fit_transform(documents).todense()
    return pd.DataFrame(tfidf, columns=vect.get_feature_names(), index=index).T

print('\n\nTF-IDF: \n')
print(get_tfidf(documents, ngram_range=(1,2), index=document_names))
```

Output - TF-IDF

```
TF-IDF:

                   IMDB 1    IMDB 2    IMDB 3
alive            0.150004  0.000000  0.000000
and              0.088595  0.122451  0.122451
and must         0.150004  0.000000  0.000000
and trade        0.000000  0.157678  0.157678
at               0.000000  0.157678  0.157678
...                   ...       ...       ...
wakes up         0.150004  0.000000  0.000000
will             0.000000  0.157678  0.157678
will facilitate  0.000000  0.157678  0.157678
world            0.150004  0.000000  0.000000
world is         0.150004  0.000000  0.000000

[82 rows x 3 columns]
```