GenomikOdev2(031690058-031790058)

May 31, 2021

1 GENOMİK HESAPLAMA ÖDEVİ 2 (31.05.2021)

Danışman: Doç.Dr.Gıyasettin Özcan

Öğrenciler: - Ozan Kılınçer (031690058) - Aleyna Er (031790058)

Bursa Uludağ Üniversitesi - BMB4020

[1]:
```
# gerekli kütüphaneler import edilir

import pandas as pd
import numpy as np
import nltk

nltk.download("stopwords")
from nltk.corpus import stopwords

nltk.download("wordnet")
from nltk.stem import WordNetLemmatizer

from nltk.stem import PorterStemmer
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Asus\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Asus\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[2]:
```
addr = r"C:\Users\Asus\Desktop\compGenomic_hw2\texts\dataset\*txt"

corpus = []

import glob
import errno

files = glob.glob(addr)
for name in files:
    try:
        with open(name) as f:
            raw_text = f.read()
            corpus.append(raw_text)
            f.close()
    except IOError as exc:
        if exc.errno != errno.EISDIR:
            raise
#print(corpus)

# metinleri okuyup diziye attık
```

[ ]:
```
corpus[9]
```

[3]:
```
# okunan metinlerde nlp işlemleri uygula: kelimeleri tokenla,noktalama␣
 ↪işaretlerini kaldır ve  köklerine ayır

corpus_processed = []

for i in range(len(corpus)):
    temp_text = corpus[i]
    #print("\n ************** \n")
    #print(temp_text)

    tokenList = nltk.word_tokenize(temp_text)
    tokenList = list(filter(lambda token: nltk.tokenize.punkt.PunktToken(token).
 ↪is_non_punct, tokenList))
    tokenList = list(filter(lambda token: token not in stopwords.
 ↪words("english"), tokenList))

    #lemmatizer = WordNetLemmatizer()
    #tokenList = [lemmatizer.lemmatize(word) for word in tokenList]

    stemmer = PorterStemmer()
    tokenList = [stemmer.stem(word) for word in tokenList]

    tokenList = [word.lower() for word in tokenList]

    corpus_processed.append(tokenList)
```

[ ]:
```
print(corpus_processed)
```

[ ]:
```
corpus_processed[1]
```

[4]:
```
txts = ["A1","A2","A3","A4","A5","A6","A7","A8","A9","A10"]
  #index (0,  1,   2,   3,   4,   5,   6,   7,   8,    9)
```

```
for x in range(len(corpus_processed)):

        tokens = corpus_processed[x]
        txts[x] = pd.DataFrame(tokens, columns = ["Words"])
        txts[x].sort_values("Words", ascending = True, inplace = True,␣
 ↪ignore_index=True)
```

[5]:
```
pd.options.display.max_rows = 100
```

[6]:
```
txts[0] # sadece ilk metni yazdırdık, txts dizisi tüm metin tablolarını tutar
```

[6]:
```
       Words
0       1862
1       1887
2      2,500
3    academi
4    academi
..       …
203    vault
204      way
205     wine
206     work
207     work

[208 rows x 1 columns]
```

[7]:
```
# her metin için ayrı ayrı tf hesapla

for x in range(len(txts)):
    df = txts[x]
    term_freq = np.zeros(len(df))
    term_idf = np.zeros(len(df))
    df["word_count"] = term_freq
    df["tf"] = term_idf
    count = 0

    for i in range(len(df)):
        word = df["Words"].values[i]
        #count += 1
        #j = i+1
        for j in range(len(df)):
            nextWord = df["Words"].values[j]
            if(word == nextWord):
                count +=1
            #else:
```

```
        df["word_count"].values[i] = count # kelimenin bir metinde kaç kez␣
↪geçtiği bilgisidir
        df["tf"].values[i] = count / len(df) # kelimenin geçme sayısı/␣
↪metindeki tüm kelimelerin sayısı

        count = 0
    df.drop_duplicates(inplace=True, ignore_index=True) # tablonun güzel␣
↪gözükmesi için tekrarlayan satırları sildik
    # silme işlemi tablonun uzunluğunu bastırırken etkiler, tf hesaplarken␣
↪etkilemez

    df.set_index(np.arange(len(df)), inplace = True)
    txts[x] = df
    #print(x)
```

[8]:
```
txts[0]
```

[8]:
```
       Words  word_count        tf
0       1862         1.0  0.004808
1       1887         1.0  0.004808
2      2,500         1.0  0.004808
3    academi         2.0  0.009615
4       acid         1.0  0.004808
..       …          …        …
154   variou         1.0  0.004808
155    vault         1.0  0.004808
156      way         1.0  0.004808
157     wine         1.0  0.004808
158     work         2.0  0.009615

[159 rows x 3 columns]
```

[10]:
```
# her kelime için idf hesapla

for x in range(len(txts)):
    df = txts[x]
    doc_freq = np.zeros(len(df))
    df["idf"] = doc_freq
    doc_num = np.zeros(len(df))
    df["num_of_doc"] = doc_num
    count = 0
    #df (metin) seçtik


    for i in range(len(df)): #dfin içindeyiz kelime seçiyoruz
        word = df["Words"].values[i]
        for j in range(len(txts)): # diğer dfleri kontrol edecek o kelime var␣
↪mı diye
            ddf = txts[j]
            if(word in ddf["Words"].values):
                count +=1

        df["num_of_doc"].values[i] = count
        df["idf"].values[i] = round(10/count,2) # toplamda 10 adet doküman vardı


        count = 0
    txts[x] = df
    #print(x)
```

[11]:
```
txts[0]
```

[11]:
```
        Words  word_count        tf   idf  num_of_doc
0        1862         1.0  0.004808   5.0         2.0
1        1887         1.0  0.004808  10.0         1.0
2       2,500         1.0  0.004808  10.0         1.0
3     academi         2.0  0.009615  10.0         1.0
4        acid         1.0  0.004808   5.0         2.0
..        …           …        …     …          …
154    variou         1.0  0.004808  10.0         1.0
155     vault         1.0  0.004808   5.0         2.0
156       way         1.0  0.004808  10.0         1.0
157      wine         1.0  0.004808  10.0         1.0
158      work         2.0  0.009615   1.0        10.0

[159 rows x 5 columns]
```

[ ]:
```
A = 0 # tüm metin tabloları yazdırır

for i in range(len(txts)):
    A = A+1
    print("\n ****** \n")
    print(" Text {} : ".format(A))
    print(txts[i])
```

1.0.1 İSTENEN 2 : A1 ile A3 tf-idf değerlerini hesaplayın

[12]:
```
A1 = txts[0]
A1
```

[12]:
```
        Words  word_count        tf   idf  num_of_doc
0        1862         1.0  0.004808   5.0         2.0
1        1887         1.0  0.004808  10.0         1.0
```

```
2      2,500         1.0  0.004808  10.0         1.0
3    academi         2.0  0.009615  10.0         1.0
4       acid         1.0  0.004808   5.0         2.0
..       …          …         …    …         …
154   variou         1.0  0.004808  10.0         1.0
155    vault         1.0  0.004808   5.0         2.0
156      way         1.0  0.004808  10.0         1.0
157     wine         1.0  0.004808  10.0         1.0
158     work         2.0  0.009615   1.0        10.0

[159 rows x 5 columns]
```

[13]:
```python
A3 = txts[2]
A3
```

[13]:
```
       Words  word_count        tf    idf  num_of_doc
0         's         9.0  0.019780   1.11         9.0
1         10         1.0  0.002198   2.50         4.0
2         17         1.0  0.002198   5.00         2.0
3       1864         1.0  0.002198  10.00         1.0
4       1867         1.0  0.002198  10.00         1.0
..       …          …         …     …         …
300     york         1.0  0.002198  10.00         1.0
301    young         1.0  0.002198  10.00         1.0
302    youth         1.0  0.002198  10.00         1.0
303  zweiten         1.0  0.002198  10.00         1.0
304     über         1.0  0.002198   5.00         2.0

[305 rows x 5 columns]
```

2 Inverted Index Oluşturun

[14]:
```python
#concat -> inverted index oluştur, metin tablolarını birleştirerek

inv_index = pd.DataFrame()

for df in range(len(txts)):
    x = txts[df].copy()
    x.drop(columns = ["tf"], inplace = True)
    inv_index = pd.concat([inv_index, x], axis = 0, ignore_index=True)

inv_index
```

[14]:
```
            Words  word_count    idf  num_of_doc
0            1862         1.0   5.00         2.0
1            1887         1.0  10.00         1.0
```

```
2         2,500          1.0  10.00          1.0
3       academi          2.0  10.00          1.0
4          acid          1.0   5.00          2.0
…           …         …    …          …
2202       work          1.0   1.00         10.0
2203 world-lead          1.0  10.00          1.0
2204    written          1.0  10.00          1.0
2205       year          3.0   1.25          8.0
2206  zoologist          1.0  10.00          1.0

[2207 rows x 4 columns]
```

[15]:
```
cols = ["A1","A2","A3","A4","A5","A6","A7","A8","A9","A10"]

for i in range(len(cols)):
    col_name = cols[i]
    col = np.zeros(len(inv_index))
    inv_index[col_name] = col
    #print(col_name)
```

[16]:
```
inv_index # boş sütunlar eklendi
```

[16]:
```
           Words  word_count    idf  num_of_doc   A1   A2   A3   A4   A5   A6  \
0           1862         1.0   5.00         2.0  0.0  0.0  0.0  0.0  0.0  0.0
1           1887         1.0  10.00         1.0  0.0  0.0  0.0  0.0  0.0  0.0
2          2,500         1.0  10.00         1.0  0.0  0.0  0.0  0.0  0.0  0.0
3        academi         2.0  10.00         1.0  0.0  0.0  0.0  0.0  0.0  0.0
4           acid         1.0   5.00         2.0  0.0  0.0  0.0  0.0  0.0  0.0
…            …          …     …           …    …    …    …    …    …    …
2202        work         1.0   1.00        10.0  0.0  0.0  0.0  0.0  0.0  0.0
2203  world-lead         1.0  10.00         1.0  0.0  0.0  0.0  0.0  0.0  0.0
2204     written         1.0  10.00         1.0  0.0  0.0  0.0  0.0  0.0  0.0
2205        year         3.0   1.25         8.0  0.0  0.0  0.0  0.0  0.0  0.0
2206   zoologist         1.0  10.00         1.0  0.0  0.0  0.0  0.0  0.0  0.0

       A7   A8   A9  A10
0     0.0  0.0  0.0  0.0
1     0.0  0.0  0.0  0.0
2     0.0  0.0  0.0  0.0
3     0.0  0.0  0.0  0.0
4     0.0  0.0  0.0  0.0
…     …    …    …    …
2202  0.0  0.0  0.0  0.0
2203  0.0  0.0  0.0  0.0
2204  0.0  0.0  0.0  0.0
2205  0.0  0.0  0.0  0.0
2206  0.0  0.0  0.0  0.0
```

```
[2207 rows x 14 columns]
```

[17]:
```
# her kelime için inverted index oluştur

for i in range(len(inv_index)): #dfin içindeyiz kelime seçiyoruz
    word = inv_index["Words"].values[i]
    cur_col = 4
    for j in range(len(txts)): # diğer dfleri kontrol edecek o kelime var mı␣
 ↪diye
        ddf = txts[j]
        ccol = inv_index.columns[cur_col]
        if(word in ddf["Words"].values):

            inv_index[ccol].values[i] = 1
            #print(ccol)
        cur_col += 1
```

[18]:
```
inv_index.drop(["word_count"], axis=1, inplace=True)
inv_index.drop(["idf"], axis=1, inplace=True) # bu iki bilgi inverted index'te␣
 ↪gerekli değil
inv_index

# kelimenin bulunduğu dokümanlar 1, olmayanlar 0 değerini alır
```

[18]:
```
            Words  num_of_doc   A1   A2   A3   A4   A5   A6   A7   A8   A9  A10
0            1862         2.0  1.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0
1            1887         1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2           2,500         1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3         academi         1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4            acid         2.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
…             …         …   …   …   …   …   …   …   …   …   …   …
2202         work        10.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
2203   world-lead         1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
2204      written         1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
2205         year         8.0  0.0  1.0  1.0  1.0  1.0  1.0  1.0  0.0  1.0  1.0
2206    zoologist         1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0

[2207 rows x 12 columns]
```

3 Sorgu metni (A11) en çok hangi metne benziyor

[19]:
```
# sorgu metni okunur ve aynı nlp işlemlerinden geçer

query_addr = r"C:\Users\Asus\Desktop\compGenomic_hw2\texts\query\A11.txt"
with open(query_addr) as f:
        query_text = f.read()
        tokens = nltk.word_tokenize(query_text)
        tokens = list(filter(lambda token: nltk.tokenize.punkt.
 ↪PunktToken(token).is_non_punct, tokens))
        tokens = list(filter(lambda token: token not in stopwords.
 ↪words("english"), tokens))

        lemmatizer = WordNetLemmatizer()
        tokens = [lemmatizer.lemmatize(word) for word in tokens]

        stemmer = PorterStemmer()
        tokens = [stemmer.stem(word) for word in tokens]

        tokens = [word.lower() for word in tokens]
        f.close()

query_df = pd.DataFrame(tokens, columns = ["words"])
query_df.sort_values("words", ascending = True, inplace = True,␣
 ↪ignore_index=True)
query_df
```

[19]:
```
    words
0  comput
1  physic
2  scienc
3  theori
```

3.1 Metinlerin benzerlik oranı aşağıdaki formülle hesaplanmıştır

(bkz. Introduction to Probabilistic Models for Information Retrieval - Victor Lavrenko / University of Edinburgh)

[20]:
```
# A11 en çok hangisine benziyor bulunur

doc_scores = {} # tüm metinlerin A11'e benzerlik puanını tutar

total_score = 1
max_score = 0
N = 10 # number of documents
constant = 0.5
```

```
for x in range(len(txts)): # df seçtik
    df = txts[x]
    cur_doc = cols[x]

    for i in range(len(df)): #dfin içindeyiz kelime seçiyoruz
        word = df["Words"].values[i]
         # diğer dfleri kontrol edecek o kelime var mı diye
        if(word in query_df["words"].values):
            print("text is:", cur_doc ,"word is:", word,"location is:", i)
            word_docnum = df["num_of_doc"].values[i]

            word_score = float((N - word_docnum + constant) / word_docnum +␣
 ↪constant)
            total_score *= word_score
            #print("word score is", word_score)

    if(total_score == 1):
        total_score = 0 # eğer iki metin ortak kelime barındırmıyorsa tatal␣
 ↪skor 0
        print("ortak kelime yok")

    total_score = round(total_score,2)
    doc_scores[cur_doc] = total_score
    print("similarity score is :", total_score)
    print("\n")

    if(total_score > max_score):
        max_score = total_score

    total_score = 1 # farklı bir metne geçerken 1'e eşitlenir
```

```
text is: A1 word is: scienc location is: 134
text is: A1 word is: theori location is: 145
similarity score is : 2.56


text is: A2 word is: physic location is: 101
text is: A2 word is: theori location is: 138
similarity score is : 3.4


text is: A3 word is: physic location is: 195
text is: A3 word is: theori location is: 258
similarity score is : 3.4
```

```
ortak kelime yok
similarity score is : 0


text is: A5 word is: comput location is: 78
text is: A5 word is: scienc location is: 199
text is: A5 word is: theori location is: 221
similarity score is : 7.68


text is: A6 word is: comput location is: 50
text is: A6 word is: scienc location is: 186
text is: A6 word is: theori location is: 208
similarity score is : 7.68


ortak kelime yok
similarity score is : 0


text is: A8 word is: comput location is: 39
text is: A8 word is: physic location is: 107
similarity score is : 6.38


text is: A9 word is: physic location is: 179
text is: A9 word is: scienc location is: 211
similarity score is : 3.4


text is: A10 word is: scienc location is: 135
similarity score is : 1.6
```

[21]:
```
doc_scores # görülüyor ki sorgu metni (A11) en çok A5 ve A6'ya benziyor
```

[21]:
```
{'A1': 2.56,
 'A2': 3.4,
 'A3': 3.4,
 'A4': 0,
 'A5': 7.68,
 'A6': 7.68,
 'A7': 0,
 'A8': 6.38,
 'A9': 3.4,
 'A10': 1.6}
```