chap2023.py

```
# cos類似度と極性分類の確信度から類似文書を表示する。知識情報演習3に独自の工夫を追加。
# クエリに対して文書のスコアを出すプログラム(0 <= SCORE <= 1)


import math
import pandas as pd
import re
from janome.tokenizer import Tokenizer
from asari.api import Sonar
import os
from IPython.terminal.prompts import Token
import stopwords


sonar = Sonar()
t = Tokenizer()

query = '日本 ワールドカップ'
#query = 'ナゲッツ ファイナル'
#query = 'NBA トレード'
#query = '渡邊雄太 NBA'
#query = 'ウェンバンヤマ ドラフト'
argv = query.split()
argc = len(argv)

# クエリ表示
print(argv)
# ストップワードリストを取得
stoplist = stopwords.stopwords()

# cos類似度を計算 ----------------------------------
def cos(argv):
    index_file = "index/index.txt"

    idf_scores = {}
    tfidf_scores = {}
    query_tf = {}
    query_tfidf = {}
    query_words = {}
    ranking_docs = {}

    #tfidf_scoresからデータフレームを作成
    f = open(index_file, 'r')

    for line in f:
        line = line.rstrip()
        split_line = line.split('\t')
        word = split_line[0]
        doc = split_line[1]
        idf = float(split_line[2])
        tfidf = float(split_line[3])

        if word not in tfidf_scores:
            tfidf_scores[word] = {}
            tfidf_scores[word][doc] = tfidf
        else:
```

```
57              tfidf_scores[word][doc] = tfidf
58
59          if word not in idf_scores:
60              idf_scores[word] = {}
61              idf_scores[word] = idf
62          else:
63              idf_scores[word] = idf
64
65      tfidf_table = pd.DataFrame(tfidf_scores)
66      tfidf_table = tfidf_table.fillna(0)
67
68
69      #クエリに対する処理
70      #argv = '吾輩は猫である'
71      query_file = 'query'
72
73      pattern = re.compile(r"^[ -—]$")
74
75
76      #tokens = t.tokenize(argv[1])
77      tokens = []
78      for i in range(argc):
79          list = t.tokenize(argv[i])
80          tokens.append(list)
81      flat_t = [item for sublist in tokens for item in sublist]
82
83      flat_token = []
84      for token in flat_t:
85          tmp = token.surface
86          judge = pattern.match(tmp)
87          flat_token.append(tmp)
88
89          if judge:
90              continue
91          if tmp in stoplist:
92              continue
93
94          if tmp in query_words:
95              query_words[tmp] += 1
96          else:
97              query_words[tmp] = 1
98
99
100
101     #クエリtfidf_scoresからクエリのデータフレームを作る
102     for index_word in idf_scores:
103         query_tf[index_word] = {}
104         query_tfidf[index_word] = {}
105         query_tf[index_word][query_file] = 0
106         query_tfidf[index_word][query_file] = 0
107
108     for query_word in query_words:
109         if query_word in query_tf:
110             query_tf[query_word][query_file] += 1
111
112     for query_word in query_words:
113         if query_word in query_tf:
114             query_tfidf[query_word][query_file] = float(query_tf[query_word]
    [query_file]) * float(idf_scores[query_word])
115
```

```
116     query_table = pd.DataFrame(query_tfidf)
117 
118     #クエリが索引後辞書の中でどの文書名に属すのかを確認し、対象文書を同定
119     for query_word in query_words:
120         if query_word in tfidf_scores:
121             for doc in tfidf_scores[query_word]:
122                 ranking_docs[doc] = 1
123 
124     #cos類似度計算
125     for doc in ranking_docs:
126         numerator = 0
127         doc_vec = tfidf_table.loc[doc]
128         query_vec = query_table.loc[query_file]
129         for i in range(len(query_vec.values)):
130             i_value = query_vec.values[i] * doc_vec.values[i]
131             numerator = numerator + i_value
132 
133         denominator = 0
134         query_value = 0
135         doc_value = 0
136         for i in range(len(query_vec.values)):
137             query_value += query_vec.values[i]**2
138             doc_value += doc_vec.values[i]**2
139 
140         denominator = math.sqrt(query_value) * math.sqrt(doc_value)
141         cosine = numerator / denominator
142 
143         ranking_docs[doc] = cosine
144 
145     return ranking_docs.items() #sorted(ranking_docs.items(), key=lambda x:x[1],
    reverse=True)
146 
147 
148 
149 # 文書のポジティブ/ネガティブを変数scaledで0から1で表示 ------------------------------
150 def text_emotions():
151     data = []
152     for filename in os.listdir('text'):
153         doc_box = []
154         f = open('text/' + filename, 'r')
155         for row in f:
156             row = row.rstrip()
157             doc_box.append(row)
158         result_doc = ''.join(doc_box)
159         res = sonar.ping(text=result_doc)
160         classes = res['classes']
161         positive_confidence = next((item['confidence'] for item in classes if
    item['class_name'] == 'positive'), 0)
162         negative_confidence = next((item['confidence'] for item in classes if
    item['class_name'] == 'negative'), 0)
163 
164         scaled = (positive_confidence - negative_confidence + 1) / 2
165         data.append(scaled)
166     return data
167 
168 
169 # クエリのポジティブ/ネガティブを変数scaledで-1から1で表示 ------------------------------
170 def query_emotions(argv):
171     query = ' '.join(argv)
172     res = sonar.ping(text=query)
173     classes = res['classes']
```

```
    positive_confidence = next((item['confidence'] for item in classes if item['class_name'] == 'positive'), 0)
    negative_confidence = next((item['confidence'] for item in classes if item['class_name'] == 'negative'), 0)
    scaled = (positive_confidence - negative_confidence + 1) / 2
    return scaled



# ポジネガの検索式を定義 ----------------------------------------
# クエリと文書のポジネガが違かったら、距離を調整(0.5倍)
def dis_emotions(qemo, temo):
    if qemo >= 0.5:
        if temo >= 0.5:
            dis = abs(qemo - temo)
        else:
            dis = abs(qemo - temo) * 0.5
    elif qemo < 0.5:
        if temo <= 0.5:
            dis = abs(qemo - temo)
        else:
            dis = abs(qemo - temo) * 0.5
    scaled = 1 - dis*2
    return scaled



# 検索スコアのためのアルゴリズムを定義 ----------------
# score = cos類似度*1/3 + ポジネガ確信度*1/3
# 0 < score < 1 に調整
def score(cos, emo):
    return cos*(2/3) + emo*(1/3)




# クエリと文書のポジネガの距離を計算 -----------
te = text_emotions()
qe = query_emotions(argv)

emo_dis = []
counter = 1
for i in te:
    tmp = dis_emotions(qe, i)
    emo_dis.append(['doc{}'.format(counter),tmp])
    counter += 1


# 表示 ----------------------------------------
score_pair = []
cosinfo = cos(argv)
coslist = list(cosinfo)
for i in range(len(cosinfo)):
    doc_score = score(coslist[i][1], emo_dis[i][1])
    score_pair.append(['doc{} '.format(i+1), doc_score])

for i in sorted(score_pair, key=lambda x: x[1], reverse=True):
    print(i)
```