

chap2023.py

```
1 # 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。
2 # クエリに対して文書のスコアを出すプログラム(0 <= SCORE <= 1)
3
4
5 import math
6 import pandas as pd
7 import re
8 from janome.tokenizer import Tokenizer
9 from asari.api import Sonar
10 import os
11 from IPython.terminal.prompts import Token
12 import stopwords
13
14
15 sonar = Sonar()
16 t = Tokenizer()
17
18 #query = 'モラント 心配'
19 #query = 'ウォリアーズ 放出'
20 #query = 'プレイオフ ファイナル'
21 #query = 'W杯 日本'
22 query = 'オフシーズン ドラフト'
23
24
25 argv = query.split()
26 argc = len(argv)
27
28 # クエリ表示
29 print(argv)
30 # ストップワードリストを取得
31 stoplist = stopwords.stopwords()
32
33 # cos類似度を計算 -----
34 def cos(argv):
35     index_file = "index/index.txt"
36
37     idf_scores = {}
38     tfidf_scores = {}
39     query_tf = {}
40     query_tfidf = {}
41     query_words = {}
42     ranking_docs = {}
43
44     #tfidf_scoresからデータフレームを作成
45     f = open(index_file, 'r')
46
47     for line in f:
48         line = line.rstrip()
49         split_line = line.split('\t')
50         word = split_line[0]
51         doc = split_line[1]
52         idf = float(split_line[2])
53         tfidf = float(split_line[3])
54
55         if word not in tfidf_scores:
56             tfidf_scores[word] = {}
```

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

```

116     query_tfidf[query_word][query_file] = float(query_tf[query_word]
117 [query_file]) * float(idf_scores[query_word])
118
119
120 #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
121 for query_word in query_words:
122     if query_word in tfidf_scores:
123         for doc in tfidf_scores[query_word]:
124             ranking_docs[doc] = 1
125
126 #cos類似度計算
127 for doc in ranking_docs:
128     numerator = 0
129     doc_vec = tfidf_table.loc[doc]
130     query_vec = query_table.loc[query_file]
131     for i in range(len(query_vec.values)):
132         i_value = query_vec.values[i] * doc_vec.values[i]
133         numerator = numerator + i_value
134
135     denominator = 0
136     query_value = 0
137     doc_value = 0
138     for i in range(len(query_vec.values)):
139         query_value += query_vec.values[i]**2
140         doc_value += doc_vec.values[i]**2
141
142     denominator = math.sqrt(query_value) * math.sqrt(doc_value)
143     cosine = numerator / denominator
144
145     ranking_docs[doc] = cosine
146
147 return ranking_docs.items() #sorted(ranking_docs.items(), key=lambda x:x[1],
reverse=True)
148
149
150
151 # 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 -----
152 def text_emotions():
153     data = []
154     for filename in os.listdir('text'):
155         doc_box = []
156         f = open('text/' + filename, 'r')
157         for row in f:
158             row = row.rstrip()
159             doc_box.append(row)
160         result_doc = ''.join(doc_box)
161         res = sonar.ping(text=result_doc)
162         classes = res['classes']
163         positive_confidence = next((item['confidence'] for item in classes if
item['class_name'] == 'positive'), 0)
164         negative_confidence = next((item['confidence'] for item in classes if
item['class_name'] == 'negative'), 0)
165
166         scaled = (positive_confidence - negative_confidence + 1) / 2
167         data.append(scaled)
168     return data
169
170
171 # 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 -----
172 def query_emotions(argv):

```

```
173     query = ' '.join(argv)
174     res = sonar.ping(text=query)
175     classes = res['classes']
176     positive_confidence = next((item['confidence'] for item in classes if
177 item['class_name'] == 'positive'), 0)
178     negative_confidence = next((item['confidence'] for item in classes if
179 item['class_name'] == 'negative'), 0)
180     scaled = (positive_confidence - negative_confidence + 1) / 2
181     return scaled
182
183 # ポジネガの検索式を定義 -----
184 # 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)
185 def dis_emotions(qemo, temo):
186     if qemo >= 0.5:
187         if temo >= 0.5:
188             dis = abs(qemo - temo)
189         else:
190             dis = abs(qemo - temo) * 0.5
191     elif qemo < 0.5:
192         if temo <= 0.5:
193             dis = abs(qemo - temo)
194         else:
195             dis = abs(qemo - temo) * 0.5
196     scaled = 1 - dis*2
197     return scaled
198
199
200
201 # 検索スコアのためのアルゴリズムを定義 -----
202 # score = cos類似度*1/3 + ポジネガ確信度*1/3
203 # 0 < score < 1 に調整
204 def score(cos, emo):
205     return cos*(2/3) + emo*(1/3)
206
207
208
209
210 # クエリと文書のポジネガの距離を計算 -----
211 te = text_emotions()
212 qe = query_emotions(argv)
213
214 emo_dis = []
215 counter = 1
216 for i in te:
217     tmp = dis_emotions(qe, i)
218     emo_dis.append(['doc{}'.format(counter), tmp])
219     counter += 1
220
221
222 # 表示 -----
223 score_pair = []
224 cosinfo = cos(argv)
225 coslist = list(cosinfo)
226 for i in range(len(cosinfo)):
227     doc_score = score(coslist[i][1], emo_dis[i][1])
228     score_pair.append(['独自スコア: doc{}'.format(i+1), doc_score])
229
230 for i in sorted(score_pair, key=lambda x: x[1], reverse=True):
```

```
231|     print(i)
```