

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



دانشکده مهندسی برق و کامپیوتر

بازیابی هوشمند اطلاعات - تمرین پنجم

سید مهدی رضوی

استاد : خانم دکتر شاکری

دی ماه ۱۴۰۲

فهرست مطالب

۳	۱ تمرین اول
۳	۱.۱ پیش‌پردازش بر روی ستون داده متنی
۵	۲.۱ job1
۷	۳.۱ job2
۹	۴.۱ job3
۱۲	۵.۱ job4
۱۴	۲ تمرین دوم
۱۸	۳ تمرین سوم

فهرست تصاویر

۶	۱ نتایج حاصل از اجرای کد برای شمارش میزان نظرات تایید شده و تایید نشده
۸	۲ استفاده از اندیس بازخورد برای ایجاد یک لیست شاخص معکوس برای کلمات موجود در بازخوردها
۱۰	۳ کلماتی که در بیشترین بازخورد ظاهر شده‌اند.
۱۳	۴ کلماتی که در بیشترین نظرات تایید شده ظاهر شده‌اند.
۱۴	۵ کلماتی که در بیشترین نظرات تایید نشده ظاهر شده‌اند.
۱۵	۶ PageRank Scores Histogram
۱۵	۷ Authority Scores Histogram
۱۶	۸ Hub Scores Histogram
۱۶	۹ میزان اشتراک در بین نقاط در میان رویکردهای امتیازدهی
۱۷	۱۰ میزان اشتراک در بین نقاط در میان رویکردهای امتیازدهی
۱۸	۱۱ رابطه هموارساز لاپلاسی برای جلوگیری از وجود احتمال صفر برای یک کلمه

۱ تمرین اول

۱.۱ پیش پردازش بر روی ستون داده متنی

```
import nltk
nltk.download('popular')

import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

def tokenize(column):
    tokens = nltk.word_tokenize(column)
    return [w for w in tokens if w.isalpha()]

def remove_stopwords(tokenized_column):
    stops = set(stopwords.words("english"))
    return [word for word in tokenized_column if not word in stops]

def apply_stemming(tokenized_column):
    stemmer = PorterStemmer()
    return [stemmer.stem(word) for word in tokenized_column]
```

در کد بالا عملیات Tokenization ، حذف کلمات یا عبارات StopWords و در نهایت عملیات Stemming را مشاهده می فرمایید.
ما ستون ReviewBody را داریم و در کنار آن ستون را می سازیم و سپس ReviewBodyPreProcess را از روی آن خواهیم ساخت.

```
data['stopwords_removed'] = data.apply(lambda x: remove_stopwords(x['tokenized']),
                                       axis=1)
data[['ReviewBody', 'stopwords_removed']].head()

"""# Stemming"""

data['porter_stemmed'] = data.apply(lambda x: apply_stemming(x['stopwords_removed']),
                                    axis=1)
# data[['Review', 'porter_stemmed']].head()

data[['ReviewBody', 'porter_stemmed']].tail()

data['ReviewBodyPreProcess'] = data['porter_stemmed']

# data.insert(7 , 'RemovedStopWords' , '')

# data['RemovedStopWords'] = data.apply(lambda x: remove_stopwords(x['ReviewBody']),
                                       axis=1)

# data.head()

data.drop('RemovedStopWords' , axis = 1)

"""# Exporting from dataframe"""

data.to_csv("BA_AirlineReviews_output.csv" , index = False)
```

```
import csv
import io
from mrjob.job import MRJob

class BooleanColumnMR(MRJob):
    def configure_args(self):
        super(BooleanColumnMR, self).configure_args()
        self.add_file_arg('--database')

    def mapper(self, _, line):
        byte_line = line.encode('utf-8') # Encode the line to bytes
        text_line = byte_line.decode('utf-8') # Decode the byte line to text
        file_like = io.StringIO(text_line) # Create a file-like object from the text
        line
        reader = csv.reader(file_like)

        for row in reader:
            # print(f'len row : {len(row)} \n')
            if len(row) > 6:
                # splitted = row.split(',')
                value = row[5].lower() # Assuming the 6th column is at index 5 (0-based
                    index)

                if value == 'true':
                    yield 'True', 1
                elif value == 'false':
                    yield 'False', 1

    def reducer(self, key, values):
        yield key, sum(values)

if __name__ == '__main__':
    job = BooleanColumnMR(args=['--database', 'BA_AirlineReviews.csv'])
    # print('Im here !!!')
    job.run()

    # CountFalseTrue.run()
```

```
File Edit View Search Terminal Help
mahdi@MahdiRazavi: ~/Documents/bazyabi/HW5/Exercise1
mahdi@MahdiRazavi:~/Documents/bazyabi/HW5/Exercise1$ python3 job_1.py /home/mahdi/Documents/bazyabi/HW5/Exercise1/BA_AirlineReviews_output.csv
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/job_1.mahdi.20231231.200113.963125
Running step 1 of 1...
job output is in /tmp/job_1.mahdi.20231231.200113.963125/output
Streaming final output from /tmp/job_1.mahdi.20231231.200113.963125/output...
"True" 1140
"False" 2547
Removing temp directory /tmp/job_1.mahdi.20231231.200113.963125...
mahdi@MahdiRazavi:~/Documents/bazyabi/HW5/Exercise1$
```

شکل ۱: نتایج حاصل از اجرای کد برای شمارش میزان نظرات تایید شده و تایید نشده

در این تمرین در قسمت Mapper در هر سطر هر یک از دو مقدار Boolean اتفاق افتاده در آن سطر و ستون مدنظر را به عدد یک نظیر می‌کنیم. سپس در گام Reducer میزان جمع هر کدام از این مقادیر Boolean را محاسبه خواهیم کرد. در واقع این گام یک Aggregator برای ما خواهد بود. تابع Aggregator ما Sum خواهد بود.

```
import csv
from mrjob.job import MRJob

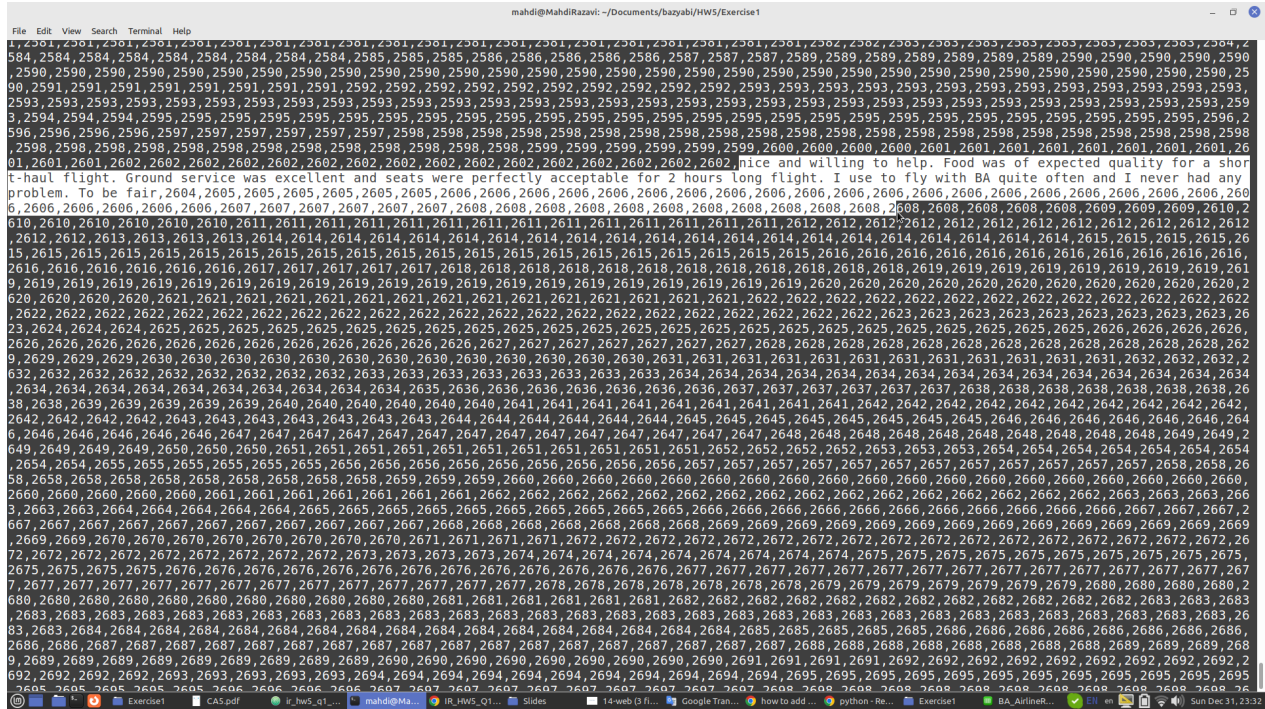
class InvertedIndex(MRJob):

    def mapper(self, _, line):
        reader = csv.reader([line])
        for row in reader:
            if len(row) > 8 :
                text = row[7]
                # Assuming the textual column is the senventh column
                # words = text.split()
                for word in text:
                    yield word, row[0] # Emitting the word as the key and the row index
                                     as the value

    def reducer(self, word, row_indices):
        yield word, ','.join(row_indices)

if __name__ == '__main__':
    InvertedIndex.run()
```

در گام Mapper می‌بایستی هر کلمه موجود را به کد سند (ایندکس) نظیر کنیم.
سپس در گام Reducer تابع Aggregator ما در واقع Join بر روی همه ایندکس‌ها خواهد بود



شکل ۲: استفاده از اندیس بازخورد برای ایجاد یک لیست شاخص معکوس برای کلمات موجود در بازخوردها


```
from mrjob.job import MRJob
import csv

class TopWordsMR(MRJob):
    def configure_args(self):
        super(TopWordsMR, self).configure_args()
        self.add_file_arg('--database')

    def mapper(self, _, line):
        # Split the line into fields using CSV reader
        row = next(csv.reader([line]))

        if len(row) > 8:
            # feedback_index = int(row[0]) # Assuming the feedback index is the first
            # field
            feedback = row[7] # Assuming the feedback text is the second field

            words = set(feedback.split(',')) # Split the feedback into words and create
            # a set

            for word in words:
                yield word, 1

    def reducer_init(self):
        self.top_words = []

    def reducer(self, word, counts):
        total_count = sum(counts)
        self.top_words.append((word, total_count))
        self.top_words = sorted(self.top_words, key=lambda x: x[1], reverse=True)[:5]

    def reducer_final(self):
        for word, count in self.top_words:
            yield word, count

if __name__ == '__main__':
    job = TopWordsMR(args = ['--database', 'BA_AirlineReviews_output.csv'])
    job.run()
```

```
File Edit View Search Terminal Help
mahdi@MahdiRazavi: ~/Documents/bazyabi/HWS/Exercise1
096,3696,3696,3696,3696,3696,3697,3697,3697,3698,3698,3698,3698,3698,3698,3698,3699,3700,3700,3700,3700,3700"
Removing temp directory /tmp/job_2.mahdi.20231231.200208.039646...
mahdi@MahdiRazavi:~/Documents/bazyabi/HWS/Exercise1$
mahdi@MahdiRazavi:~/Documents/bazyabi/HWS/Exercise1$ python3 job_3.py /home/mahdi/Documents/bazyabi/HWS/Exercise1/BA_AirlineReviews_output.csv
No configs found: falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/job_3.mahdi.20231231.200306.208134
Running step 1 of 1...
job output is in /tmp/job_3.mahdi.20231231.200306.208134/output
Streaming final output from /tmp/job_3.mahdi.20231231.200306.208134/output...
{"time": 1556,
 "would": 1050,
 "use": 780,
 "travel": 716,
 "us": 636,
 "flight": 2068,
 "food": 1627,
 "fli": 1058,
 "drink": 859,
 "economy": 851,
 "crew": 1381,
 "british": 1362,
 "cabin": 1293,
 "class": 1011,
 "busi": 985,
 "london": 1291,
 "one": 1118,
 "offer": 839,
 "meal": 736,
 "loung": 675,
 "ba": 2051,
 "airway": 1401,
 "board": 1098,
 "airlin": 1087,
 "arriv": 781,
 "return": 868,
 "passeng": 836,
 "plane": 797,
 "realli": 540,
 "poor": 496,
 "the": 2060,
 "seat": 2059,
 "servic": 1739,
 "staff": 1070,
 "serv": 668,
 "i": 2629,
 "good": 1276,
 "hour": 1181,
 "healthrow": 1143,
 "get": 1021}
Removing temp directory /tmp/job_3.mahdi.20231231.200306.208134...
mahdi@MahdiRazavi:~/Documents/bazyabi/HWS/Exercise1$
```

شکل ۳: کلماتی که در بیشترین بازخورد ظاهر شده‌اند.



```
from mrjob.job import MRJob
import csv

class TopWordsTrueMR(MRJob):
    def configure_args(self):
        super(TopWordsTrueMR, self).configure_args()
        self.add_file_arg('--database')

    def mapper(self, _, line):
        # Split the line into fields using CSV reader
        row = next(csv.reader([line]))

        if len(row) > 8:
            # feedback_index = int(row[0]) # Assuming the feedback index is the first
            # field
            feedback = row[7] # Assuming the feedback text is the second field
            column_six = row[5] # Assuming column[6] is the seventh field (index 6)

            if column_six.lower() == 'true':
                words = set(feedback.split()) # Split the feedback into words and
                # create a set

                for word in words:
                    yield word, 1

    def reducer_init(self):
        self.top_words = []

    def reducer(self, word, counts):
        total_count = sum(counts)
        self.top_words.append((word, total_count))
        self.top_words = sorted(self.top_words, key=lambda x: x[1], reverse=True)[:5]

    def reducer_final(self):
        for word, count in self.top_words:
            yield word, count

if __name__ == '__main__':
    job = TopWordsTrueMR(args=['--database', 'BA_AirlineReviews_output.csv'])
    job.run()
```

```
mahdi@MahdiRazavi: ~/Documents/bazyabi/HW5/Exercise1
File Edit View Search Terminal Help
mahdi@MahdiRazavi:~/Documents/bazyabi/HW5/Exercise1$ python3 job_4.py /home/mahdi/Documents/bazyabi/HW5/Exercise1/BA_AirlineReviews_output.csv
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/job_4.mahdi.20231231.200344.659861
Running Step 1 of 1...
job output is in /tmp/job_4.mahdi.20231231.200344.659861/output
Streaming final output from /tmp/job_4.mahdi.20231231.200344.659861/output...
'time', 464
'would', 388
'us', 206
'use', 203
'we', 196
'flight', 891
'food', 422
'fli', 290
'even', 257
'experi', 225
'crew', 388
'cabin', 338
'british', 329
'busi', 294
'class', 291
'london', 450
'one', 347
'offer', 263
'meal', 203
'like', 195
'ba', 615
'board', 332
'airway', 324
'airlin', 306
'arriv', 272
'passeng', 234
'plane', 210
'return', 181
'realli', 155
'pay', 128
'the', 632
'seat', 540
'servic', 549
'staff', 324
'they', 178
'i', 814
'hour', 381
'get', 322
'good', 311
'heathrow', 303
Removing temp directory /tmp/job_4.mahdi.20231231.200344.659861...
mahdi@MahdiRazavi:~/Documents/bazyabi/HW5/Exercise1$
```

شکل ۴: کلماتی که در بیشترین نظرات تایید شده ظاهر شده‌اند.

این گام بسیار شبیه به گام قبل می‌باشد با این تفاوت که در گام Reducer یک قید دیگر به ازای هر سطر وجود دارد که باید ستون تایید شده یا True باشد و یا این که False باشد.

```

mahdi@MahdiRazavi: ~/Documents/bazyabi/HWS/Exercise1
File Edit View Search Terminal Help
Streaming final output from /tmp/job_4.mahdi.20240101.054340.083326/output...
'would', 742
'use', 577
'travel', 520
'us', 430
'we', 429
'flight', 1074
'food', 1205
'fli', 767
'economi', 649
'drink', 639
'crew', 990
'cabin', 955
'class', 719
'busi', 688
'could', 591
'london', 840
'one', 770
'offer', 576
'meal', 533
'lounge', 506
'ba', 1433
'airway', 1075
'british', 1030
'airlin', 781
'board', 766
'return', 686
'passeng', 602
'plane', 578
'realli', 385
'poor', 368
'seat', 1507
'the', 1427
'servic', 1188
'time', 1090
'staff', 746
'i', 1813
'good', 965
'heathrow', 840
'hour', 800
'get', 690
Removing temp directory /tmp/job_4.mahdi.20240101.054340.083326...
mahdi@MahdiRazavi:~/Documents/bazyabi/HWS/Exercise1$

```

شکل ۵: کلماتی که در بیشترین نظرات تایید نشده ظاهر شده‌اند.

۲ تمرین دوم

الگوریتم PageRank در واقع به صورت زیر عمل می‌کند که وزن (رتبه) عددی را به هر گره در نمودار اختصاص می‌دهد که نشان دهنده اهمیت یا اعتبار آن است.

رتبه را بر اساس تعداد و کیفیت لینک‌های ورودی به یک گره محاسبه می‌کند.

مرتباً رتبه‌ها را تا رسیدن به همگرایی به روز می‌کند.

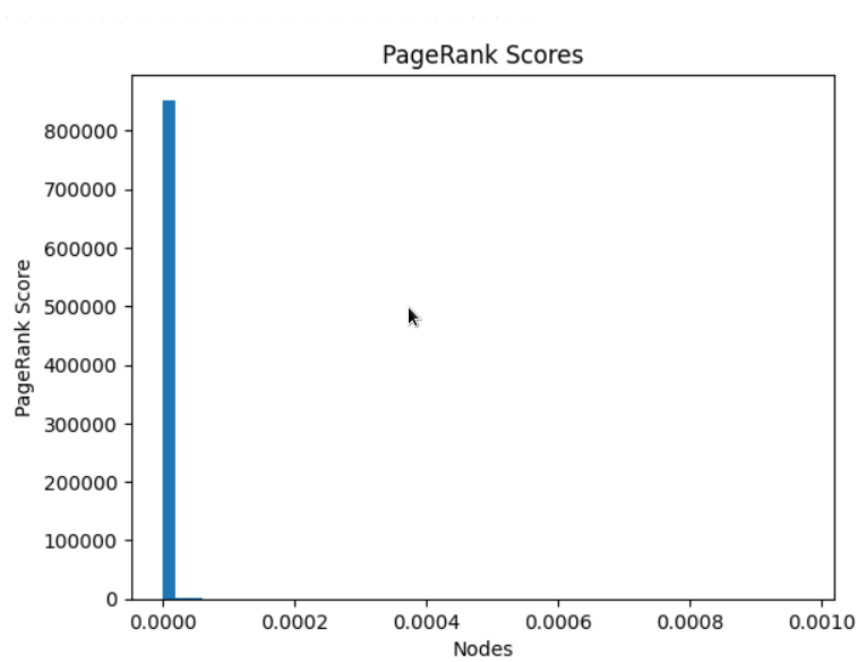
اما رویکرد الگوریتم Hub به این صورت است که دو نوع گره را شناسایی می‌کند: مقامات (منابع محتوای با کیفیت بالا) و هاب (صفحاتی که به بسیاری از مقامات پیوند دارند). امتیازات اعتبار را بر اساس تعداد و کیفیت لینک‌های دریافتی از هاب محاسبه می‌کند. امتیازات هاب را بر اساس تعداد و کیفیت پیوندهای خروجی به مقامات محاسبه می‌کند. به طور مکرر امتیازها را تا رسیدن به همگرایی به روز می‌کند.

۱۰۰۰ گره مشترک بین این دو الگوریتم صفحاتی را نشان می‌دهد که هم بسیار معتبر هستند و هم به خوبی متصل است این صفحات احتمالاً مرتبط هستند، زیرا هم توسط منابع معتبر دیگر و هم صفحاتی که به بسیاری از منابع معتبر پیوند دارند تأیید شده‌اند. PageRank همه پیوندهای ورودی را به طور مساوی در نظر می‌گیرد، در حالی که HITS بین پیوندها از هاب و غیرهاب تمایز قائل می‌شود.

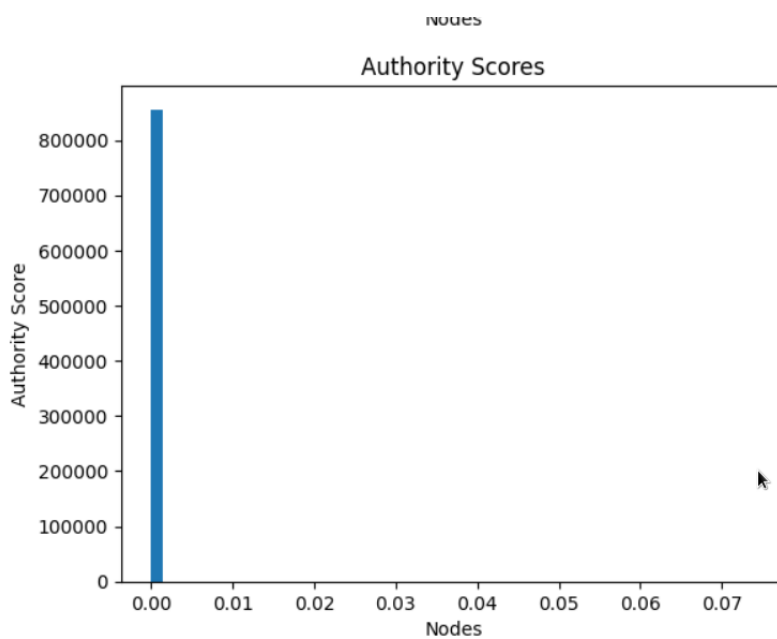
الگوریتم PageRank با در نظر گرفتن کل ساختار به صورت global تر عمل می‌کند در حالی که HITS محلی‌تر عمل می‌کند و بر همسایگی گره تمرکز دارد.

اندازه تعداد رئوس بزرگترین مولفه همبندی ضعیف این گراف : ۸۵۵۸۰۲

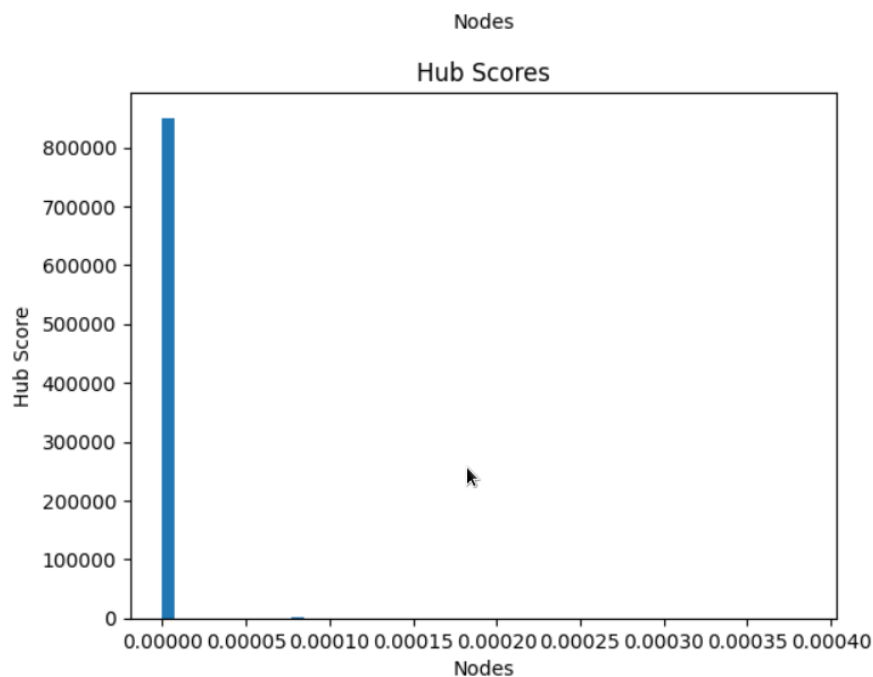
اندازه تعداد یال‌های رئوس بزرگترین مولفه همبندی ضعیف این گراف : ۵۰۶۶۸۴۱



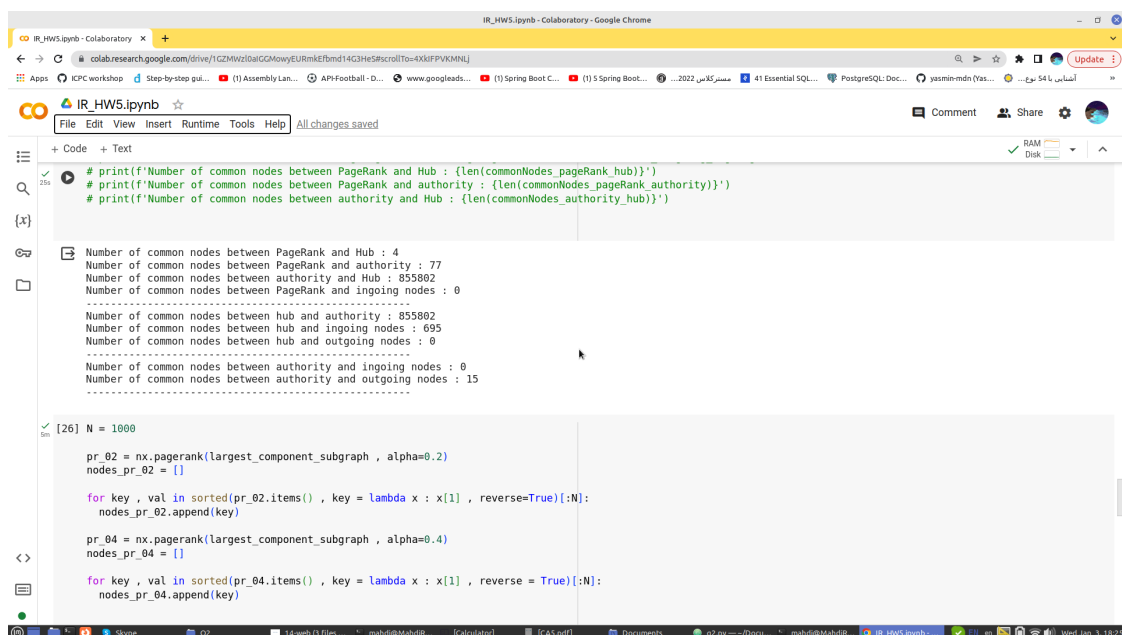
شکل ۶: PageRank Scores Histogram



شکل ۷: Authority Scores Histogram



شکل ۸: Hub Scores Histogram



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
# print(f'Number of common nodes between PageRank and Hub : {len(commonNodes_pageRank_hub)}')
```

```
# print(f'Number of common nodes between PageRank and authority : {len(commonNodes_pageRank_authority)}')
```

```
# print(f'Number of common nodes between authority and Hub : {len(commonNodes_authority_hub)}')
```

Number of common nodes between PageRank and Hub : 4
Number of common nodes between PageRank and authority : 77
Number of common nodes between authority and Hub : 855802
Number of common nodes between PageRank and ingoing nodes : 0
.....
Number of common nodes between hub and authority : 855802
Number of common nodes between hub and ingoing nodes : 695
Number of common nodes between hub and outgoing nodes : 0
.....
Number of common nodes between authority and ingoing nodes : 0
Number of common nodes between authority and outgoing nodes : 15
.....

```
[26] N = 1000
```

```
pr_02 = nx.pagerank(largest_component_subgraph , alpha=0.2)
```

```
nodes_pr_02 = []
```

```
for key , val in sorted(pr_02.items() , key = lambda x : x[1] , reverse=True)[:N]:
```

```
nodes_pr_02.append(key)
```

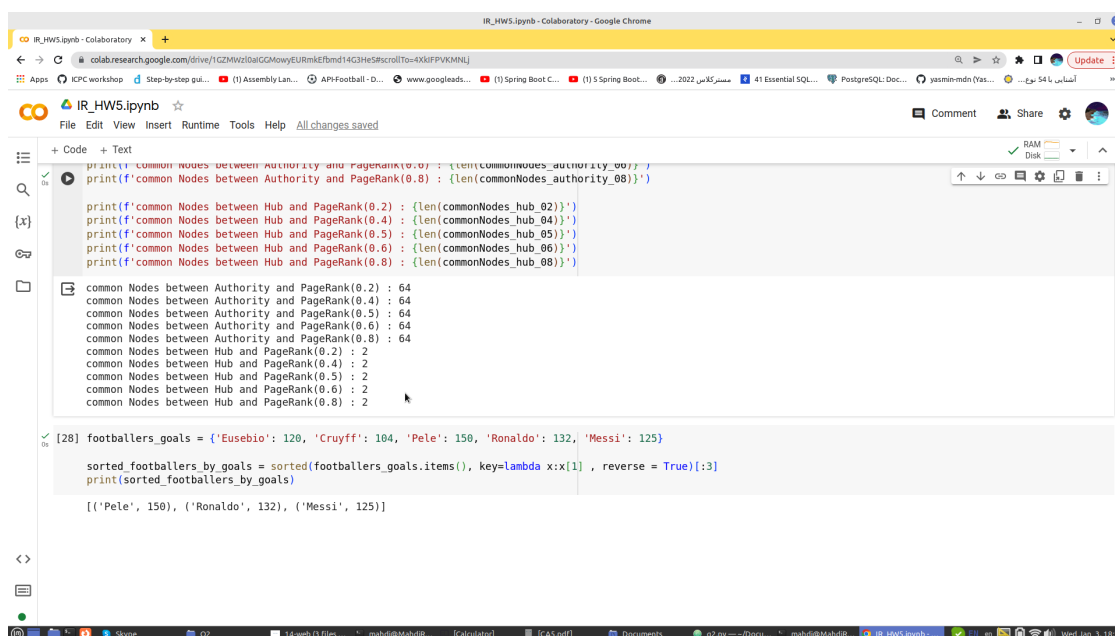
```
pr_04 = nx.pagerank(largest_component_subgraph , alpha=0.4)
```

```
nodes_pr_04 = []
```

```
for key , val in sorted(pr_04.items() , key = lambda x : x[1] , reverse = True)[:N]:
```

```
nodes_pr_04.append(key)
```

شکل ۹: میزان اشتراک در بین نقاط در میان رویکردهای امتیازدهی



```
IR_HWS.ipynb - Colaboratory
colabresearch.google.com/drive/1GZMvz10arGGMoyEUKmkEfmd14G3He5fscrollTo=4XkIFPVKMNL
Apps
ICPC workshop
Step-by-step gui...
(1) Assembly Lan...
API-Football - D...
www.googleads...
(1) Spring Boot C...
(1) 5 Spring Boot...
مسترکلاس 2022
41 Essential SQL...
PostgreSQL Doc...
yamin-mdn (yas...
امشاس 54 نوع...
Update

IR_HWS.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
print('common Nodes between Authority and PageRank(0.0) : {len(commonNodes_authority_00)}')
print('common Nodes between Authority and PageRank(0.8) : {len(commonNodes_authority_08)}')

print(f'common Nodes between Hub and PageRank(0.2) : {len(commonNodes_hub_02)}')
print(f'common Nodes between Hub and PageRank(0.4) : {len(commonNodes_hub_04)}')
print(f'common Nodes between Hub and PageRank(0.5) : {len(commonNodes_hub_05)}')
print(f'common Nodes between Hub and PageRank(0.6) : {len(commonNodes_hub_06)}')
print(f'common Nodes between Hub and PageRank(0.8) : {len(commonNodes_hub_08)}')

common Nodes between Authority and PageRank(0.2) : 64
common Nodes between Authority and PageRank(0.4) : 64
common Nodes between Authority and PageRank(0.5) : 64
common Nodes between Authority and PageRank(0.6) : 64
common Nodes between Authority and PageRank(0.8) : 64
common Nodes between Hub and PageRank(0.2) : 2
common Nodes between Hub and PageRank(0.4) : 2
common Nodes between Hub and PageRank(0.5) : 2
common Nodes between Hub and PageRank(0.6) : 2
common Nodes between Hub and PageRank(0.8) : 2

[28] footballers_goals = {'Eusebio': 120, 'Cruyff': 104, 'Pele': 150, 'Ronaldo': 132, 'Messi': 125}

sorted_footballers_by_goals = sorted(footballers_goals.items(), key=lambda x:x[1], reverse = True)[:3]
print(sorted_footballers_by_goals)

[('Pele', 150), ('Ronaldo', 132), ('Messi', 125)]
```

شکل ۱۰: میزان اشتراک در بین نقاط در میان رویکردهای امتیازدهی

۳ تمرین سوم

ما برای الگوریتم Expect Maximization روابط زیر را در اختیار داریم :

$$P(d) = P(\Theta_1)P(d|\Theta_1) + P(\Theta_2)P(d|\Theta_2)$$

$$P(d|\Lambda) = \sum_{i=1}^k (P(\Theta_i) \prod_{w \in V} P(w|\Theta_i)^{c(w,d)})$$

$$\Lambda^* = \operatorname{argmax}_{\Lambda} P(d|\Lambda)$$

هموارسازی لاپلاسی الگوریتمی برای هموار کردن یک شبکه چند ضلعی است. برای هر راس در یک مش، یک موقعیت جدید بر اساس اطلاعات محلی (مانند موقعیت همسایگان) انتخاب می شود و راس به آنجا منتقل می شود. در صورتی که یک مش از نظر توپولوژیکی یک شبکه مستطیل شکل باشد (یعنی هر رأس داخلی به چهار همسایه متصل است) سپس این عملیات لاپلاسی مش را تولید می کند.

میزان احتمال وجود یک ترم خاص در یک Cluster به صورت هموارسازی لاپلاس محاسبه شده است. با توجه به این که در هر Cluster ما ۳ ترم منحصربفرد داریم ، تعداد گره های ما ۳ تا خواهد بود. به ازای هموارسازی با فاکتور ۱ خواهیم داشت :

$$P(A|Cluster1) = \frac{4+1}{9+3}$$

$$\begin{aligned} \hat{P}(w_i | c) &= \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} (\text{count}(w, c) + 1)} \\ &= \frac{\text{count}(w_i, c) + 1}{\left(\sum_{w \in V} \text{count}(w, c) \right) + |V|} \end{aligned}$$

شکل ۱۱: رابطه هموارسازی لاپلاسی برای جلوگیری از وجود احتمال صفر برای یک کلمه

$$P(\Theta_1) = \frac{3}{4}$$

$$P(\Theta_2) = \frac{1}{4}$$

$$P(A|Cluster1) = \frac{5}{12}$$

$$P(B|Cluster1) = \frac{3}{12}$$

$$P(C|Cluster1) = \frac{1}{12}$$

$$P(D|Cluster1) = \frac{4}{12}$$

$$P(A|Cluster2) = \frac{2}{8}$$

$$P(B|Cluster2) = \frac{3}{8}$$

$$P(C|Cluster2) = \frac{3}{8}$$

$$P(D|Cluster2) = \frac{1}{8}$$

$$P(Document1|Cluster1) = \frac{3}{4} * (\frac{5}{12})^2 * \frac{3}{12} = 0.032552083(X1)$$

$$P(Document1|Cluster2) = \frac{1}{4} * (\frac{2}{8})^2 * \frac{3}{8} = 0.005859375(Y1)$$

Document1 Assigned to Cluster1

$$P(Document2|Cluster1) = \frac{3}{4} * (\frac{3}{12})^2 * (\frac{1}{12})^2 * \frac{5}{12} = 0.000135634(X2)$$

$$P(Document2|Cluster2) = \frac{1}{4} * (\frac{3}{8})^2 * (\frac{3}{8})^2 * \frac{2}{8} = 0.001235962(Y2)$$

Document2 Assigned to Cluster2

$$P(Document3|Cluster1) = \frac{3}{4} * \frac{3}{12} * \frac{5}{12} * (\frac{4}{12})^2 = 0.008680556(X3)$$

$$P(Document3|Cluster2) = \frac{1}{4} * \frac{3}{8} * \frac{2}{8} * (\frac{1}{8})^2 = 0.000366211(Y3)$$

Document3 Assigned to Cluster1

$$P(Document4|Cluster1) = \frac{3}{4} * \frac{5}{12} * \frac{4}{12} = 0.104166667(X4)$$

$$P(Document4|Cluster2) = \frac{1}{4} * \frac{2}{8} * \frac{1}{8} = 0.0078125(Y4)$$

Document4 Assigned to Cluster1

مرحله به روزرسانی میزان احتمال هریک از خوشه ها :

$$P(\Theta_1) = \frac{X1 + X2 + X3 + X4}{4} = 0.364$$

$$P(\Theta_2) = \frac{Y1 + Y2 + Y3 + Y4}{4} = 0.003825$$

$$\text{Manhattan} - \text{Dist}(D1, C1) = 3$$

$$\text{Manhattan} - \text{Dist}(D1, C2) = 4$$

Document1 Assigned to C1.

$$\text{Manhattan} - \text{Dist}(D2, C1) = 5$$

$$\text{Manhattan} - \text{Dist}(D2, C2) = 2$$

Document2 Assigned to C2

$$\text{Manhattan} - \text{Dist}(D3, C1) = 2$$

$$\text{Manhattan} - \text{Dist}(D3, C2) = 5$$

Document3 Assigned to C1

$$\text{Manhattan} - \text{Dist}(D4, C1) = 0$$

$$\text{Manhattan} - \text{Dist}(D4, C2) = 5$$

Document4 Assigned to C1

K-Means Clustering

$$Document5 = CAA$$

$$Manhattan - Dist(D5, C1) = 3$$

$$Manhattan - Dist(D5, C2) = 4$$

Document5 Assigned to C1

EM generative Model

$$Document5 = CAA$$

$$P(Document5|Cluster1) = P(\Theta_1) * P(C|Cluster1) * P(A|Cluster1)^2 = 0.005266204$$

$$P(Document5|Cluster2) = P(\Theta_2) * P(C|Cluster2) * P(A|Cluster2)^2 = 0.000008965$$

Document5 Assigned to C1

هر دو رویکرد الگوریتمی داکيومنت ۵ را به کلاستر اول نظیر خواهد کرد.

EM-Generative و K-Means هر دو الگوریتم‌های خوشه‌بندی هستند که از آن‌ها استفاده زیادی می‌شود. مزایای EM-Generative :

۱. می‌تواند داده‌های پراکنده و دارای توزیع نامناسب ، مانند مثال همین سوال را تا حد بسیار خوبی بر اساس احتمال Background مناسب ، خوشه‌بندی کند.
۲. می‌تواند خوشه‌هایی با اشکال ، اندازه‌ها و چگالی متفاوت تولید کند.
۳. EM-Generative می‌تواند داده‌های از دست رفته را با وارد کردن مقادیر گم‌شده با استفاده از مدلی که از داده‌های موجود به دست می‌آید، مدیریت کند.
۴. EM-Generative می‌تواند برای شناسایی الگوها در داده‌های با ابعاد بالا با کاهش ابعاد داده‌ها با استفاده از تکنیک‌هایی استفاده شود.

معایب EM-Generative:

۱. تولید EM می‌تواند از نظر محاسباتی محاسباتی گران باشد ، به خصوص زمانی که با مجموعه داده‌های بزرگ یا ابعاد بالا سروکار داریم.
۲. EM-Generative نیاز به تنظیم دستی پارامترهای اولیه دارد که اگر دانش قبلی در مورد ساختار داده‌ها وجود نداشته باشد ، می‌تواند چالش برانگیز باشد . انتخاب پارامترهای اولیه نیز می‌تواند بر خوشه‌های حاصل تاثیر بگذارد.
- EM ممکن است به یک بهینه محلی همگرا شود یا در یک حالت گیرکند اگر در طول فرآیند تکرار به دقت نظارت و تطبیق داده نشود.
۳. EM-Generative فرض می‌کند که هر خوشه تابع چگالی خاصی دارد ، که ممکن است همیشه در سناریوهای دنیای واقعی صادق نباشد .
- در چنین مواردی ممکن است نتایج کمتر از حد بهینه ایجاد کند یا به درستی همگرا نشود.

مزایای K-Means :

۱. k-means از نظر محاسباتی کارآمد است و پیاده‌سازی آن آسان است و آن را به یک انتخاب درست برای خوشه‌بندی مجموعه داده‌های بزرگ تبدیل می‌کند.
۲. K-Means خوشه‌هایی با ابعاد کروی و تقریباً میزان داده برابر تولید می‌کند که در بسیاری از کاربردهایی که این مفروضات درست هستند.
۳. K-Means امکان پردازش موازی را فراهم می‌کند، که می‌تواند زمان محاسبات را در هنگام برخورد با مجموعه داده‌های بزرگ یا چندین پردازنده/هسته موجود در یک ماشین/خوشه، به میزان قابل توجهی افزایش دهد.

همانطور که پیش‌تر هم ذکر شد ، رویکرد EM-Generative برای این توزیع داده بسیار مناسب‌تر خواهد بود . به شرطی که میزان احتمال هر خوشه ، بر اساس دانش قبلی مناسب به وجود بیاید.

از توزیع هندسی این نقاط ، می‌توان این نتیجه را گرفت که مدل ساده K-Means قادر به خوشه‌بندی مناسبی برای این داده‌ها نخواهد بود ، چرا که میزان شباهت بین داده‌ها اگر بر اساس فاصله اقلیدسی یا هر نوع فاصله ، دیگری باشد قادر به ارائه خوشه‌های مناسبی نخواهد بود.