homework2(1)

2024 年 4 月 22 日

1.YELP: https://www.yelp.com/dataset/download

读取数据并进行预处理

[1]:
```
import os
import pandas as pd
import json
for dirname, _, filenames in os.walk('yelp_dataset-2'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
yelp_dataset-2\Dataset_User_Agreement.pdf
yelp_dataset-2\yelp_academic_dataset_business.json
yelp_dataset-2\yelp_academic_dataset_checkin.json
yelp_dataset-2\yelp_academic_dataset_review.json
yelp_dataset-2\yelp_academic_dataset_tip.json
yelp_dataset-2\yelp_academic_dataset_user.json
```

[2]:
```
def load_rows(filepath, nrows = None):
    with open(filepath) as json_file:
        count = 0
        objs = []
        line = json_file.readline()
        while (nrows is None or count < nrows) and line:
            count += 1
            obj = json.loads(line)
            objs.append(obj)
            line = json_file.readline()
        return pd.DataFrame(objs)
```

[3]:
```
review = load_rows('yelp_dataset-2/yelp_academic_dataset_review.json',10000)
review.head(10)
```

[3]:
```
               review_id                 user_id             business_id  \
0  KU_O5udG6zpxOg-VcAEodg  mh_-eMZ6K5RLWhZyISBhwA  XQfwVwDr-v0ZS3_CbbE5Xw
1  BiTunyQ73aT9WBnpR9DZGw  OyoGAe7OKpv6SyGZT5g77Q  7ATYjTIgM3jUlt4UM3IypQ
2  saUsX_uimxRlCVr67Z4Jig  8g_iMtfSiwikVnbP2etR0A  YjUWPpI6HXG530lwP-fb2A
3  AqPFMleE6RsU23_auESxiA  _7bHUi9Uuf5__HHc_Q8guQ  kxX2SOes4o-D3ZQBkiMRfA
4  Sx8TMOWLNuJBWer-0pcmoA  bcjbaE6dDog4jkNY91ncLQ  e4Vwtrqf-wpJfwesgvdgxQ
5  JrIxlS1TzJ-iCu79ul40cQ  eUta8W_HdHMXPzLBBZhL1A  04UD14gamNjLY0IDYVhHJg
6  6AxgBCNX_PNTOxmbRSwcKQ  r3zeYsv1XFBRA4dJpL78cw  gmjsEdUsKpj9Xxu6pdjH0g
7  _ZeMknuYdlQcUqng_Im3yg  yfFzsLmaWF2d4Sr0UNbBgg  LHSTtnW3YHCeUkRDGyJOyw
8  ZKvDG2sBvHVdF5oBNUOpAQ  wSTuiTk-sKNdcFyprzZAjg  B5XSoSG3SfvQGtKEGQ1tSQ
9  pUycOfUwM8vqX7KjRRhUEA  59MxRhNVhU9MYndMkz0wtw  gebiRewfieSdtt17PTW6Zg

   stars  useful  funny  cool  \
0    3.0       0      0     0
1    5.0       1      0     1
2    3.0       0      0     0
3    5.0       1      0     1
4    4.0       1      0     1
5    1.0       1      2     1
6    5.0       0      2     0
7    5.0       2      0     0
8    3.0       1      1     0
9    3.0       0      0     0

                                                text                 date
0  If you decide to eat here, just be aware it is…  2018-07-07 22:09:11
1  I've taken a lot of spin classes over the year…  2012-01-03 15:28:18
2  Family diner. Had the buffet. Eclectic assortm…  2014-02-05 20:30:30
3  Wow!  Yummy, different,  delicious.   Our favo…  2015-01-04 00:01:03
4  Cute interior and owner (?) gave us tour of up…  2017-01-14 20:54:15
5  I am a long term frequent customer of this est…  2015-09-23 23:10:31
6  Loved this tour! I grabbed a groupon and the p…  2015-01-03 23:21:18
7  Amazingly amazing wings and homemade bleu chee…  2015-08-07 02:29:16
8  This easter instead of going to Lopez Lake we …  2016-03-30 22:46:33
9  Had a party of 6 here for hibachi. Our waitres…  2016-07-25 07:31:06
```

[4]:
```python
business = load_rows('yelp_dataset-2/yelp_academic_dataset_business.json',10000)
business.head(10)
```

[4]:
```
             business_id                      name  \
0  Pns2l4eNsfO8kk83dixA6A  Abby Rappoport, LAC, CMQ
1  mpf3x-BjTdTEA3yCZrAYPw            The UPS Store
2  tUFrWirKiKi_TAnsVWINQQ                   Target
3  MTSW4McQd7CbVtyjqoe9mw       St Honore Pastries
4  mWMc6_wTdE0EUBKIGXDVfA  Perkiomen Valley Brewery
5  CF33F8-E6oudUQ46HnavjQ           Sonic Drive-In
6  n_0UpQx1hsNbnPUSlodU8w          Famous Footwear
7  qkRM_2X51Yqxk3btlwAQIg           Temple Beth-El
8  k0hlBqXX-Bt0vf1op7Jr1w    Tsevi's Pub And Grill
9  bBDDEgkFA1Otx9Lfe7BZUQ           Sonic Drive-In

                                      address           city state  \
0                       1616 Chapala St, Ste 2  Santa Barbara    CA
1              87 Grasso Plaza Shopping Center         Affton    MO
2                         5255 E Broadway Blvd         Tucson    AZ
3                                  935 Race St   Philadelphia    PA
4                                101 Walnut St     Green Lane    PA
5                                615 S Main St   Ashland City    TN
6  8522 Eager Road, Dierbergs Brentwood Point      Brentwood    MO
7                          400 Pasadena Ave S  St. Petersburg    FL
8                           8025 Mackenzie Rd         Affton    MO
9                         2312 Dickerson Pike      Nashville    TN

  postal_code   latitude   longitude  stars  review_count  is_open  \
0       93101  34.426679 -119.711197    5.0             7        0
1       63123  38.551126  -90.335695    3.0            15        1
2       85711  32.223236 -110.880452    3.5            22        0
3       19107  39.955505  -75.155564    4.0            80        1
4       18054  40.338183  -75.471659    4.5            13        1
5       37015  36.269593  -87.058943    2.0             6        1
6       63144  38.627695  -90.340465    2.5            13        1
7       33707  27.766590  -82.732983    3.5             5        1
8       63123  38.565165  -90.321087    3.0            19        0
9       37207  36.208102  -86.768170    1.5            10        1

                                      attributes  \
0                   {'ByAppointmentOnly': 'True'}
1          {'BusinessAcceptsCreditCards': 'True'}
```

```
2  {'BikeParking': 'True', 'BusinessAcceptsCredit…
3  {'RestaurantsDelivery': 'False', 'OutdoorSeati…
4  {'BusinessAcceptsCreditCards': 'True', 'Wheelc…
5  {'BusinessParking': 'None', 'BusinessAcceptsCr…
6  {'BusinessAcceptsCreditCards': 'True', 'Restau…
7                                               None
8  {'Caters': 'True', 'Alcohol': 'u'full_bar'', '…
9  {'RestaurantsAttire': ''casual'', 'Restaurants…

                                          categories  \
0  Doctors, Traditional Chinese Medicine, Naturop…
1  Shipping Centers, Local Services, Notaries, Ma…
2  Department Stores, Shopping, Fashion, Home & G…
3  Restaurants, Food, Bubble Tea, Coffee & Tea, B…
4                         Brewpubs, Breweries, Food
5  Burgers, Fast Food, Sandwiches, Food, Ice Crea…
6  Sporting Goods, Fashion, Shoe Stores, Shopping…
7               Synagogues, Religious Organizations
8  Pubs, Restaurants, Italian, Bars, American (Tr…
9  Ice Cream & Frozen Yogurt, Fast Food, Burgers,…

                                               hours
0                                               None
1  {'Monday': '0:0-0:0', 'Tuesday': '8:0-18:30', …
2  {'Monday': '8:0-22:0', 'Tuesday': '8:0-22:0', …
3  {'Monday': '7:0-20:0', 'Tuesday': '7:0-20:0', …
4  {'Wednesday': '14:0-22:0', 'Thursday': '16:0-2…
5  {'Monday': '0:0-0:0', 'Tuesday': '6:0-22:0', '…
6  {'Monday': '0:0-0:0', 'Tuesday': '10:0-18:0', …
7  {'Monday': '9:0-17:0', 'Tuesday': '9:0-17:0', …
8                                               None
9  {'Monday': '0:0-0:0', 'Tuesday': '6:0-21:0', '…
```

预处理，查看是否有 null 值，若有，则删除 null 值

[5]:
```
review.isna().any(axis=1).sum()
```

[5]:
```
0
```

[6]:
```
business.isna().any(axis=1).sum()
```

[6]: 2105

[7]:
```
business_new = business.drop(['business_id','name','hours','postal_code'],1).
↪dropna()
business_new.head()
```

```
C:\Users\Superstar\AppData\Local\Temp\ipykernel_20788\3772500510.py:1:
FutureWarning: In a future version of pandas all arguments of DataFrame.drop
except for the argument 'labels' will be keyword-only
  business_new =
business.drop(['business_id','name','hours','postal_code'],1).dropna()
```

[7]:
```
                           address           city state   latitude  \
0           1616 Chapala St, Ste 2  Santa Barbara    CA  34.426679
1  87 Grasso Plaza Shopping Center         Affton    MO  38.551126
2             5255 E Broadway Blvd         Tucson    AZ  32.223236
3                      935 Race St   Philadelphia    PA  39.955505
4                    101 Walnut St     Green Lane    PA  40.338183

    longitude  stars  review_count  is_open  \
0 -119.711197    5.0             7        0
1  -90.335695    3.0            15        1
2 -110.880452    3.5            22        0
3  -75.155564    4.0            80        1
4  -75.471659    4.5            13        1

                                          attributes  \
0                        {'ByAppointmentOnly': 'True'}
1                {'BusinessAcceptsCreditCards': 'True'}
2  {'BikeParking': 'True', 'BusinessAcceptsCredit…
3  {'RestaurantsDelivery': 'False', 'OutdoorSeati…
4  {'BusinessAcceptsCreditCards': 'True', 'Wheelc…

                                          categories
0  Doctors, Traditional Chinese Medicine, Naturop…
1  Shipping Centers, Local Services, Notaries, Ma…
2  Department Stores, Shopping, Fashion, Home & G…
3  Restaurants, Food, Bubble Tea, Coffee & Tea, B…
4                          Brewpubs, Breweries, Food
```

频繁模式挖掘

[8]:
```
business_new['stars'] = business_new['stars'].apply(lambda x: str(int(x //1 * 1)).
↪split())
business_new['review_count_frequency'] = business_new['review_count'].apply(lambda␣
↪x:'high_counts'.split() if x > 30 else 'low_counts'.split())
business_new['latitude'] = business_new['latitude'].apply(lambda x: str(int(x//5␣
↪*5)).split())
business_new['longitude'] = business_new['longitude'].apply(lambda x: str(int(x//5␣
↪*5)).split())
business_new['categories'] = business_new['categories'].apply(lambda x: x.split(',␣
↪'))
business_new.head()
```

[8]:
```
                           address           city state latitude longitude  \
0           1616 Chapala St, Ste 2  Santa Barbara    CA     [30]    [-120]
1  87 Grasso Plaza Shopping Center         Affton    MO     [35]     [-95]
2             5255 E Broadway Blvd         Tucson    AZ     [30]    [-115]
3                      935 Race St   Philadelphia    PA     [35]     [-80]
4                    101 Walnut St     Green Lane    PA     [40]     [-80]

  stars  review_count  is_open  \
0   [5]             7        0
1   [3]            15        1
2   [3]            22        0
3   [4]            80        1
4   [4]            13        1

                                          attributes  \
0                      {'ByAppointmentOnly': 'True'}
1               {'BusinessAcceptsCreditCards': 'True'}
2  {'BikeParking': 'True', 'BusinessAcceptsCredit…
3  {'RestaurantsDelivery': 'False', 'OutdoorSeati…
4  {'BusinessAcceptsCreditCards': 'True', 'Wheelc…

                                          categories review_count_frequency
0  [Doctors, Traditional Chinese Medicine, Naturo…           [low_counts]
1  [Shipping Centers, Local Services, Notaries, M…           [low_counts]
2  [Department Stores, Shopping, Fashion, Home & …           [low_counts]
3  [Restaurants, Food, Bubble Tea, Coffee & Tea, …          [high_counts]
```

```
4                         [Brewpubs, Breweries, Food]           [low_counts]
```

[9]:
```python
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori

transactions = list(business_new['stars'])
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df = pd.DataFrame(te_ary, columns=te.columns_)

# 使用 apriori 算法找频繁项集，设置最小支持度阈值为 0.1
frequent_itemset = apriori(df, min_support=0.1, use_colnames=True)
frequent_itemset
```

[9]:
```
    support itemsets
0  0.152845      (2)
1  0.295295      (3)
2  0.410175      (4)
3  0.105470      (5)
```

[10]:
```python
transactions_2 = list(business_new['categories'])
te_2 = TransactionEncoder()
te_ary_2 = te_2.fit(transactions_2).transform(transactions_2)
df_2 = pd.DataFrame(te_ary_2, columns=te_2.columns_)
frequent_itemset_2 = apriori(df_2, min_support=0.05, use_colnames=True)
frequent_itemset_2
```

[10]:
```
     support                          itemsets
0   0.058315                     (Active Life)
1   0.057002          (American (Traditional))
2   0.063567                      (Automotive)
3   0.079431                            (Bars)
4   0.102516                  (Beauty & Spas)
5   0.071991       (Event Planning & Services)
6   0.197593                            (Food)
7   0.081182               (Health & Medical)
8   0.092232                 (Home Services)
9   0.074836                (Local Services)
10  0.088731                       (Nightlife)
11  0.052954                           (Pizza)
```

```
12  0.375383                          (Restaurants)
13  0.059300                           (Sandwiches)
14  0.171116                             (Shopping)
15  0.057002  (Restaurants, American (Traditional))
16  0.079431                      (Nightlife, Bars)
17  0.057877                    (Restaurants, Bars)
18  0.110503                    (Restaurants, Food)
19  0.060832               (Restaurants, Nightlife)
20  0.052954                   (Pizza, Restaurants)
21  0.059300              (Restaurants, Sandwiches)
22  0.057877         (Restaurants, Bars, Nightlife)
```

[11]:
```
transactions_3 = list(business_new['latitude'])
te_3 = TransactionEncoder()
te_ary_3 = te_3.fit(transactions_3).transform(transactions_3)
df_3 = pd.DataFrame(te_ary_3, columns=te_3.columns_)
frequent_itemset_3 = apriori(df_3, min_support=0.1, use_colnames=True)
frequent_itemset_3
```

[11]:
```
    support itemsets
0  0.227899     (25)
1  0.115974     (30)
2  0.458643     (35)
3  0.159081     (40)
```

[12]:
```
transactions_4 = list(business_new['longitude'])
te_4 = TransactionEncoder()
te_ary_4 = te_4.fit(transactions_4).transform(transactions_4)
df_4 = pd.DataFrame(te_ary_4, columns=te_4.columns_)
frequent_itemset_4 = apriori(df_4, min_support=0.1, use_colnames=True)
frequent_itemset_4
```

[12]:
```
    support itemsets
0  0.104486   (-115)
1  0.113457   (-120)
2  0.253392    (-80)
3  0.173414    (-85)
4  0.166630    (-90)
5  0.142779    (-95)
```

模式命名

```
[13]: # 关联分析
from mlxtend.frequent_patterns import association_rules
rules = association_rules(frequent_itemset_2, metric="confidence", min_threshold=0.
↪01)
rules = rules[rules['lift'] > 10]
rules
```

```
[13]:               antecedents                consequents  antecedent support  \
2                 (Nightlife)                     (Bars)            0.088731
3                      (Bars)                (Nightlife)            0.079431
14        (Restaurants, Bars)                (Nightlife)            0.057877
15   (Restaurants, Nightlife)                     (Bars)            0.060832
18                     (Bars)   (Restaurants, Nightlife)            0.079431
19                (Nightlife)        (Restaurants, Bars)            0.088731

    consequent support   support  confidence       lift  leverage  conviction  \
2             0.079431  0.079431    0.895191  11.270037  0.072383    8.783311
3             0.088731  0.079431    1.000000  11.270037  0.072383         inf
14            0.088731  0.057877    1.000000  11.270037  0.052742         inf
15            0.079431  0.057877    0.951439  11.978170  0.053046   18.956901
18            0.060832  0.057877    0.728650  11.978170  0.053046    3.461098
19            0.057877  0.057877    0.652281  11.270037  0.052742    2.709438

    zhangs_metric
2        1.000000
3        0.989898
14       0.967251
15       0.975879
18       0.995596
19       1.000000
```

根据上面的关联性分析结果，给出模式命名：1. 种类差异模式：由上表分析，类别中几乎不可能同时包含 Bars 与 Nightlife，因此如果用户购买了种类包含 Bars 的产品，就尽量少推种类中包含 Nightlife 的产品了

可视化

```
[14]: new_data = business[['name', 'review_count']]
new_data = new_data.sort_values(by='review_count', ascending=False)
```

```
new_data
```

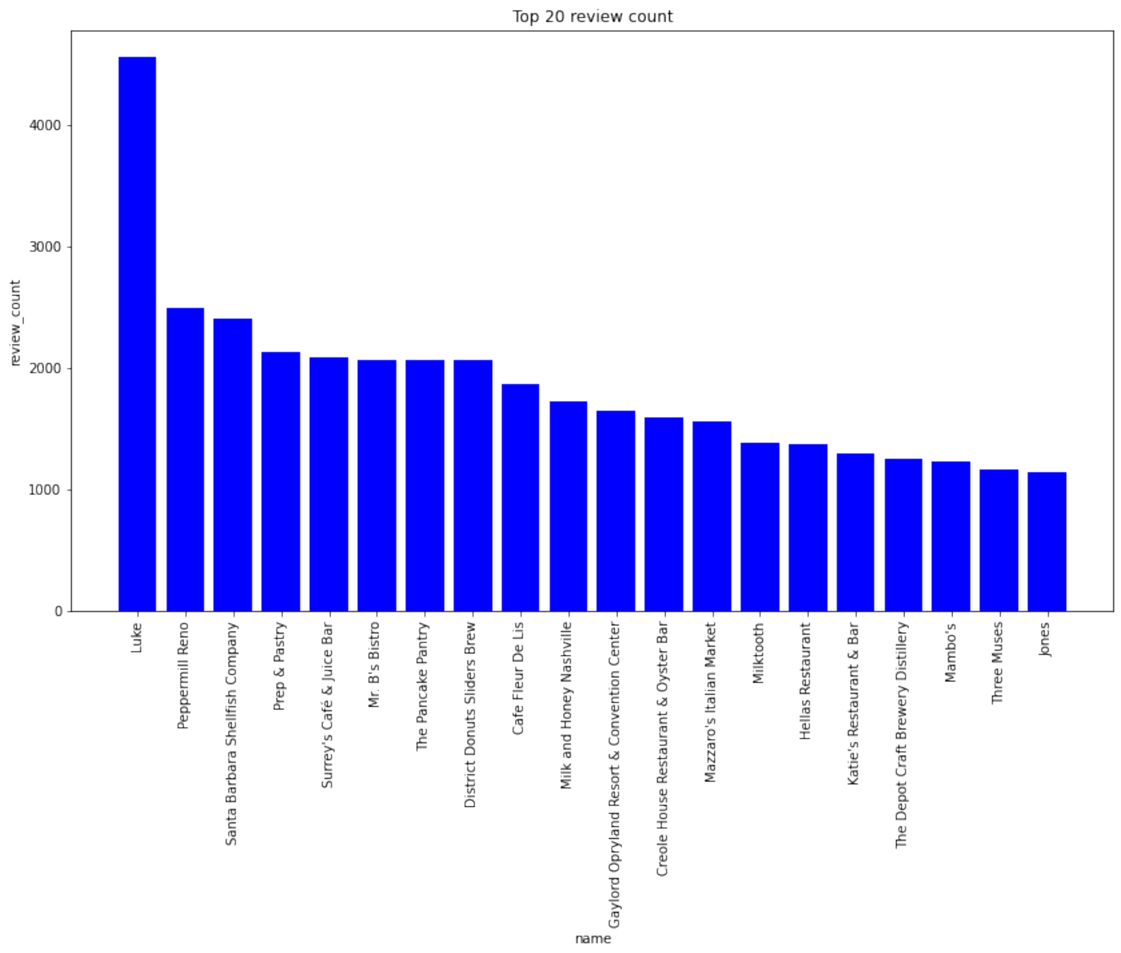
[14]:

```
                                   name  review_count
4497                               Luke          4554
9506                     Peppermill Reno         2486
141         Santa Barbara Shellfish Company      2404
2380                       Prep & Pastry         2126
5851              Surrey's Café & Juice Bar      2084
…                                   …            …
4816                         Pressuremaxx           5
9249                  Buccaneers Team Store         5
4829   National Heating & Air Conditioning          5
9252                Castleton HD Ultrasound         5
3031                        Aprils On Main          5

[10000 rows x 2 columns]
```

[15]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(14, 8))
plt.bar(new_data['name'].head(20), new_data['review_count'].head(20), color='blue')
plt.xlabel('name')
plt.ylabel('review_count')
plt.title('Top 20 review count')
plt.xticks(rotation=90)
plt.show()
```



2.Microsoft 资讯推荐: https://learn.microsoft.com/zh-cn/azure/open-datasets/dataset-microsoft-news?tabs=azureml-opendatasets

读取数据并进行预处理

[16]:
```
import tempfile
import urllib
import zipfile
temp_dir = os.path.join(tempfile.gettempdir(), 'mind')
os.makedirs(temp_dir, exist_ok=True)
base_url = 'https://mind201910small.blob.core.windows.net/release'
validation_small_url = f'{base_url}/MINDsmall_dev.zip'

def download_url(url,
                 destination_filename=None,
                 progress_updater=None,
                 force_download=False,
                 verbose=True):
    """
    Download a URL to a temporary file
    """
    if not verbose:
        progress_updater = None
    # This is not intended to guarantee uniqueness, we just know it happens to␣
↪guarantee
    # uniqueness for this application.
    if destination_filename is None:
        url_as_filename = url.replace('://', '_').replace('/', '_')
        destination_filename = \
            os.path.join(temp_dir,url_as_filename)
    if (not force_download) and (os.path.isfile(destination_filename)):
        if verbose:
            print('Bypassing download of already-downloaded file {}'.format(
                os.path.basename(url)))
        return destination_filename
    if verbose:
        print('Downloading file {} to {}'.format(os.path.basename(url),
                                                 destination_filename),
              end='')
    urllib.request.urlretrieve(url, destination_filename, progress_updater)
    assert (os.path.isfile(destination_filename))
    nBytes = os.path.getsize(destination_filename)
    if verbose:
        print('...done, {} bytes.'.format(nBytes))
    return destination_filename

zip_path = download_url(validation_small_url, verbose=True)
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(temp_dir)

os.listdir(temp_dir)
```

```
Bypassing download of already-downloaded file MINDsmall_dev.zip
```

```
[16]: ['behaviors.tsv',
       'entity_embedding.vec',
       'https_mind201910small.blob.core.windows.net_release_MINDsmall_dev.zip',
       'news.tsv',
       'relation_embedding.vec']
```

```
[17]: behaviors_path = os.path.join(temp_dir, 'behaviors.tsv')
      behaviors = pd.read_table(
          behaviors_path,
          header=None,
          names=['impression_id', 'user_id', 'time', 'history', 'impressions'])
```

```
[18]: behaviors
```

```
[18]:        impression_id user_id                   time  \
      0                  1  U80234  11/15/2019 12:37:50 PM
      1                  2  U60458   11/15/2019 7:11:50 AM
      2                  3  U44190   11/15/2019 9:55:12 AM
      3                  4  U87380   11/15/2019 3:12:46 PM
      4                  5   U9444   11/15/2019 8:25:46 AM
      …                …     …                      …
      73147          73148  U77536   11/15/2019 8:40:16 PM
      73148          73149  U56193   11/15/2019 1:11:26 PM
      73149          73150  U16799   11/15/2019 3:37:06 PM
      73150          73151   U8786   11/15/2019 8:29:26 AM
      73151          73152  U68182  11/15/2019 11:54:34 AM

                                                        history  \
      0      N55189 N46039 N51741 N53234 N11276 N264 N40716…
      1      N58715 N32109 N51180 N33438 N54827 N28488 N611…
      2      N56253 N1150 N55189 N16233 N61704 N51706 N5303…
      3      N63554 N49153 N28678 N23232 N43369 N58518 N444…
      4                     N51692 N18285 N26015 N22679 N55556
      …                                                   …
      73147  N28691 N8845 N58434 N37120 N22185 N60033 N4702…
      73148  N4705 N58782 N53531 N46492 N26026 N28088 N3109…
      73149   N40826 N42078 N15670 N15295 N64536 N46845 N52294
      73150  N3046 N356 N20483 N46107 N44598 N18693 N8254 N…
      73151  N20297 N53568 N4690 N60608 N43709 N43123 N1885…
```

```
                                         impressions
0      N28682-0 N48740-0 N31958-1 N34130-0 N6916-0 N5…
1      N20036-0 N23513-1 N32536-0 N46976-0 N35216-0 N…
2      N36779-0 N62365-0 N58098-0 N5472-0 N13408-0 N5…
3      N6950-0 N60215-0 N6074-0 N11930-0 N6916-0 N248…
4      N5940-1 N23513-0 N49285-0 N23355-0 N19990-0 N3…
…                                                 …
73147  N496-0 N35159-0 N59856-0 N13270-0 N47213-0 N26…
73148  N49285-0 N31958-0 N55237-0 N42844-0 N29862-0 N…
73149  N7043-0 N512-0 N60215-1 N45057-0 N496-0 N37055…
73150  N23692-0 N19990-0 N20187-0 N5940-0 N13408-0 N3…
73151  N29862-0 N5472-0 N21679-1 N6400-0 N53572-0 N50…

[73152 rows x 5 columns]
```

[19]:
```python
news_path = os.path.join(temp_dir, 'news.tsv')
news = pd.read_table(news_path,
              header=None,
              names=[
                  'id', 'category', 'subcategory', 'title', 'abstract', 'url',
                  'title_entities', 'abstract_entities'
              ])
news
```

[19]:
```
           id       category              subcategory  \
0      N55528      lifestyle          lifestyleroyals
1      N18955         health                  medical
2      N61837           news                newsworld
3      N53526         health                   voices
4      N38324         health                  medical
…         …            …                       …
42411  N63550      lifestyle          lifestyleroyals
42412  N30345  entertainment  entertainment-celebrity
42413  N30135           news             newsgoodnews
42414  N44276          autos              autossports
42415  N39563         sports              more_sports

                                                title  \
0      The Brands Queen Elizabeth, Prince Charles, an…
```

```
1      Dispose of unwanted prescription drugs during …
2      The Cost of Trump's Aid Freeze in the Trenches…
3      I Was An NBA Wife. Here's How It Affected My M…
4      How to Get Rid of Skin Tags, According to a De…
…                                                   …
42411  Why Kate & Meghan Were on Different Balconies …
42412           See the stars at the 2019 Baby2Baby gala
42413  Tennessee judge holds lawyer's baby as he swea…
42414                  Best Sports Car Deals for October
42415       Shall we dance: Sports stars shake their leg

                                                abstract  \
0      Shop the notebooks, jackets, and more that the…
1                                                    NaN
2      Lt. Ivan Molchanets peeked over a parapet of s…
3      I felt like I was a fraud, and being an NBA wi…
4      They seem harmless, but there's a very good re…
…                                                   …
42411  There's no scandal here. It's all about the or…
42412  Stars like Chrissy Teigen and Kate Hudson supp…
42413  Tennessee Court of Appeals Judge Richard Dinki…
42414                                                NaN
42415                                                NaN

                                                url  \
0      https://assets.msn.com/labs/mind/AAGH0ET.html
1      https://assets.msn.com/labs/mind/AAISxPN.html
2      https://assets.msn.com/labs/mind/AAJgNsz.html
3      https://assets.msn.com/labs/mind/AACk2N6.html
4      https://assets.msn.com/labs/mind/AAAKEkt.html
…                                               …
42411  https://assets.msn.com/labs/mind/BBWyynu.html
42412  https://assets.msn.com/labs/mind/BBWyz7N.html
42413  https://assets.msn.com/labs/mind/BBWyzI8.html
42414  https://assets.msn.com/labs/mind/BBy5rVe.html
42415  https://assets.msn.com/labs/mind/BBzMpnG.html

                                          title_entities  \
0      [{"Label": "Prince Philip, Duke of Edinburgh",…
```

```
1      [{"Label": "Drug Enforcement Administration", …
2                                                    []
3                                                    []
4      [{"Label": "Skin tag", "Type": "C", "WikidataI…
…                                                 …
42411  [{"Label": "Meghan, Duchess of Sussex", "Type"…
42412                                                []
42413  [{"Label": "Tennessee", "Type": "G", "Wikidata…
42414  [{"Label": "Peugeot RCZ", "Type": "V", "Wikida…
42415                                                []

                                   abstract_entities
0                                                    []
1                                                    []
2      [{"Label": "Ukraine", "Type": "G", "WikidataId…
3      [{"Label": "National Basketball Association", …
4      [{"Label": "Skin tag", "Type": "C", "WikidataI…
…                                                 …
42411                                                []
42412  [{"Label": "Kate Hudson", "Type": "P", "Wikida…
42413  [{"Label": "Tennessee Court of Appeals", "Type…
42414                                                []
42415                                                []

[42416 rows x 8 columns]
```

[20]:
```
behaviors['time'] = pd.to_datetime(behaviors['time'])
behaviors
```

[20]:
```
       impression_id user_id                time  \
0                  1  U80234 2019-11-15 12:37:50
1                  2  U60458 2019-11-15 07:11:50
2                  3  U44190 2019-11-15 09:55:12
3                  4  U87380 2019-11-15 15:12:46
4                  5   U9444 2019-11-15 08:25:46
…                …     …                 …
73147          73148  U77536 2019-11-15 20:40:16
73148          73149  U56193 2019-11-15 13:11:26
73149          73150  U16799 2019-11-15 15:37:06
```

```
73150          73151   U8786 2019-11-15 08:29:26
73151          73152  U68182 2019-11-15 11:54:34

                                                history  \
0      N55189 N46039 N51741 N53234 N11276 N264 N40716…
1      N58715 N32109 N51180 N33438 N54827 N28488 N611…
2      N56253 N1150 N55189 N16233 N61704 N51706 N5303…
3      N63554 N49153 N28678 N23232 N43369 N58518 N444…
4                     N51692 N18285 N26015 N22679 N55556
…                                                    …
73147  N28691 N8845 N58434 N37120 N22185 N60033 N4702…
73148  N4705 N58782 N53531 N46492 N26026 N28088 N3109…
73149   N40826 N42078 N15670 N15295 N64536 N46845 N52294
73150  N3046 N356 N20483 N46107 N44598 N18693 N8254 N…
73151  N20297 N53568 N4690 N60608 N43709 N43123 N1885…

                                            impressions
0      N28682-0 N48740-0 N31958-1 N34130-0 N6916-0 N5…
1      N20036-0 N23513-1 N32536-0 N46976-0 N35216-0 N…
2      N36779-0 N62365-0 N58098-0 N5472-0 N13408-0 N5…
3      N6950-0 N60215-0 N6074-0 N11930-0 N6916-0 N248…
4      N5940-1 N23513-0 N49285-0 N23355-0 N19990-0 N3…
…                                                    …
73147  N496-0 N35159-0 N59856-0 N13270-0 N47213-0 N26…
73148  N49285-0 N31958-0 N55237-0 N42844-0 N29862-0 N…
73149  N7043-0 N512-0 N60215-1 N45057-0 N496-0 N37055…
73150  N23692-0 N19990-0 N20187-0 N5940-0 N13408-0 N3…
73151  N29862-0 N5472-0 N21679-1 N6400-0 N53572-0 N50…

[73152 rows x 5 columns]
```

检查是否有 null 值，若有，将其去除

[21]:
```
behaviors.isna().any(axis=1).sum()
```

[21]:
```
2214
```

[22]:
```
behaviors = behaviors.dropna()
```

[23]:
```
behaviors['history'] = behaviors['history'].apply(lambda x: x.split(' '))
behaviors['impressions'] = behaviors['impressions'].apply(lambda x: x.split(' '))
behaviors = behaviors.head(10000)
behaviors
```

```
C:\Users\Superstar\AppData\Local\Temp\ipykernel_20788\2271076412.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  behaviors['history'] = behaviors['history'].apply(lambda x: x.split(' '))
C:\Users\Superstar\AppData\Local\Temp\ipykernel_20788\2271076412.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  behaviors['impressions'] = behaviors['impressions'].apply(lambda x: x.split('
'))
```

[23]:
```
       impression_id user_id                time  \
0                  1  U80234 2019-11-15 12:37:50
1                  2  U60458 2019-11-15 07:11:50
2                  3  U44190 2019-11-15 09:55:12
3                  4  U87380 2019-11-15 15:12:46
4                  5   U9444 2019-11-15 08:25:46
…                …     …                   …
10280          10281  U82600 2019-11-15 11:43:32
10281          10282  U33073 2019-11-15 10:34:25
10282          10283  U21980 2019-11-15 06:03:42
10283          10284  U50685 2019-11-15 11:27:28
10284          10285  U36136 2019-11-15 15:19:00

                                                 history  \
0      [N55189, N46039, N51741, N53234, N11276, N264,…
1      [N58715, N32109, N51180, N33438, N54827, N2848…
```

```
2       [N56253, N1150, N55189, N16233, N61704, N51706…
3       [N63554, N49153, N28678, N23232, N43369, N5851…
4                [N51692, N18285, N26015, N22679, N55556]
…                                                   …
10280   [N20609, N31557, N3909, N21383, N8391, N59303,…
10281   [N56253, N24724, N56586, N25415, N55808, N5518…
10282   [N11282, N52097, N1150, N11071, N62990, N64555…
10283   [N42612, N4643, N23571, N56253, N12384, N38118…
10284   [N17933, N59557, N12711, N16693, N17157, N5263…

                                              impressions
0       [N28682-0, N48740-0, N31958-1, N34130-0, N6916…
1       [N20036-0, N23513-1, N32536-0, N46976-0, N3521…
2       [N36779-0, N62365-0, N58098-0, N5472-0, N13408…
3       [N6950-0, N60215-0, N6074-0, N11930-0, N6916-0…
4       [N5940-1, N23513-0, N49285-0, N23355-0, N19990…
…                                                   …
10280   [N31958-0, N49285-0, N5472-0, N34130-0, N29862…
10281   [N51470-0, N759-0, N31958-0, N6916-0, N5472-0,…
10282   [N17807-0, N36786-0, N29490-0, N32786-0, N1968…
10283   [N29862-0, N14223-0, N34130-0, N31958-1, N5523…
10284   [N24802-0, N19611-0, N12409-0, N49285-0, N5357…

[10000 rows x 5 columns]
```

频繁模式挖掘

[24]:
```python
transactions = list(behaviors['history'])
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df = pd.DataFrame(te_ary, columns=te.columns_)

# 使用 apriori 算法找频繁项集，设置最小支持度阈值为 0.05
frequent_itemset = apriori(df, min_support=0.01, use_colnames=True)
frequent_itemset
```

[24]:
```
      support      itemsets
0      0.0426      (N10059)
1      0.0159      (N10078)
2      0.0101      (N10152)
```

```
3       0.0196         (N10235)
4       0.0153         (N10347)
…        …              …
1015    0.0100  (N871, N60184)
1016    0.0116  (N61388, N871)
1017    0.0127    (N619, N871)
1018    0.0107   (N6233, N871)
1019    0.0109   (N8148, N871)

[1020 rows x 2 columns]
```

模式命名

[28]:
```
rules = association_rules(frequent_itemset, metric="confidence", min_threshold=0.
↪001)
rules = rules[rules['lift'] > 10]
rules
```

[28]:
```
    antecedents consequents  antecedent support  consequent support  support  \
762     (N47020)    (N61864)              0.0294              0.0238   0.0119
763     (N61864)    (N47020)              0.0238              0.0294   0.0119

     confidence       lift  leverage  conviction  zhangs_metric
762    0.404762  17.006803    0.0112    1.640016       0.969709
763    0.500000  17.006803    0.0112    1.941200       0.964147
```

根据上面的关联性分析结果，给出模式命名：1. 不推荐模式：由上表分析，如果观看了新闻 N47020，就几乎不会看新闻 N61864 了，因此如果有用户观看了这两个新闻之一，就不给他推荐另外一个了。

可视化

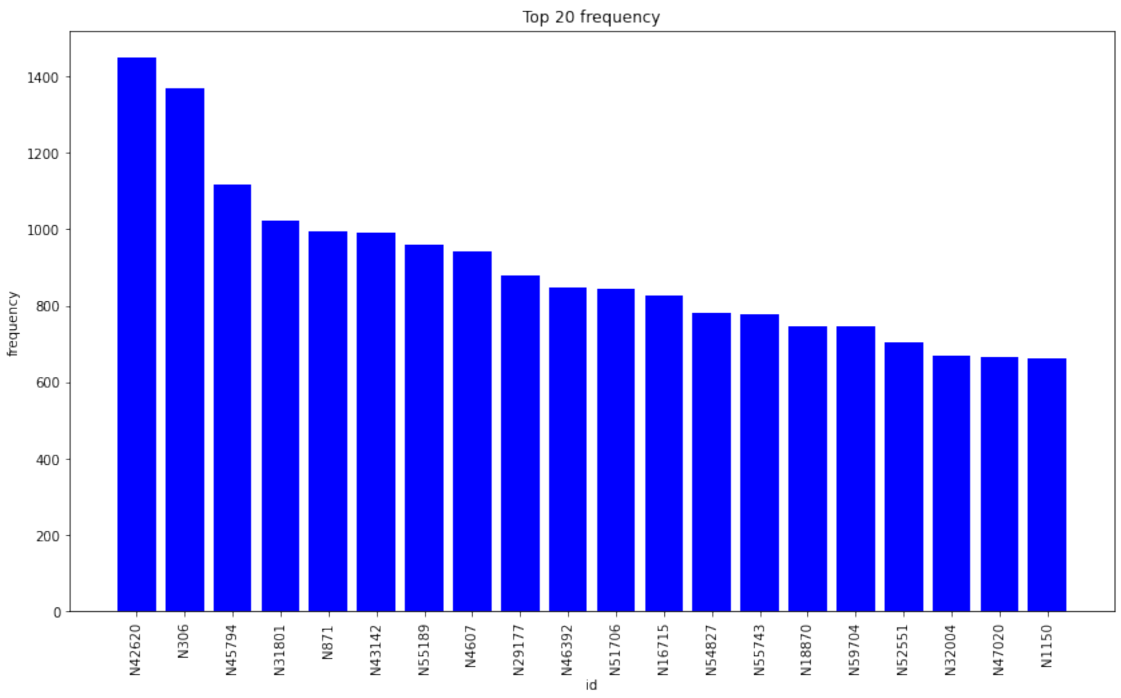
[30]:
```
from collections import Counter
new_data = [item for sublist in list(behaviors['history']) for item in sublist]
frequency = Counter(new_data)
frequency_df = pd.DataFrame(frequency.items(), columns=['id', 'frequency'])
frequency_df = frequency_df.sort_values(by='frequency', ascending=False)
frequency_df
```

[30]:
```
           id  frequency
118    N42620       1448
611      N306       1368
```

```
44      N45794       1118
238     N31801       1021
440       N871        996
…          …          …
14968    N7986          1
14964   N24232          1
14962   N18648          1
14961   N38016          1
21456   N41593          1

[21457 rows x 2 columns]
```

[31]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(14, 8))
plt.bar(frequency_df['id'].head(20), frequency_df['frequency'].head(20),␣
  ↪color='blue')
plt.xlabel('id')
plt.ylabel('frequency')
plt.title('Top 20 frequency')
plt.xticks(rotation=90)
plt.show()
```


Top 20 frequency
frequency
id
N42620
N306
N45794
N31801
N871
N43142
N55189
N4607
N29177
N46392
N51706
N16715
N54827
N55743
N18870
N59704
N52551
N32004
N47020
N1150