网页浏览行为关联规则挖掘

github地址：https://github.com/cppccp/anonymous

3220220906 崔鹏

In [9]:

```
# 数据清洗并提取用户浏览记录
file = './anonymous-msweb.data'
webid2weburl = dict()
transactions = []
for line in open(file, 'r').readlines():
    if line.startswith('A'):
        elems = line.split(',')
        webid, weburl = int(elems[1]),
elems[-1].strip('\n')
        webid2weburl[webid] = weburl
    elif line.startswith('C'):  # new transaction
begin
        transactions.append([])
    elif line.startswith('V'):
        webid = line.split(',')[1]
        transactions[-1].append(webid)

print(f"countsof transaction:
{len(transactions)}, counts of weburl:
{len(webid2weburl)}")
```

```
countsof transaction: 32711, counts of weburl: 294
```

In [32]:

```
# 分析最经常被访问的页面、页面访问量分布等
```

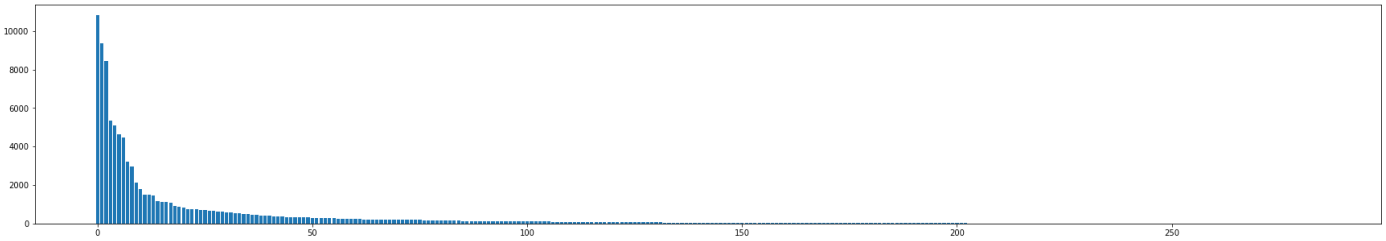
```
import numpy as np
from collections import Counter
from tqdm import tqdm

WEB_CNT = len(webid2weburl)
webid2cnt = Counter()
for transcation in tqdm(transactions,
total=len(transactions)):
    for item in transcation:
        webid2cnt[item] += 1
print("最常被访问的网址是: ")
for webid, cnt in webid2cnt.most_common(5):
    weburl = webid2weburl[webid]
    print(f'weburl: {weburl}, visit counts:
{cnt}')

# 画图分析
import matplotlib.pyplot as plt
plt.figure(figsize=(30,5))
# xticks = [webid2weburl[webid] for
webid,visit_cnt in webid2cnt.most_common()]
y = [visit_cnt for webid,visit_cnt in
webid2cnt.most_common()]
x = np.arange(len(y))
plt.bar(x, y)
# plt.xticks(x, xticks)
```

```
100%|██████████| 32711/32711 [00:00<00:00, 672436.35it/s]
最常被访问的网址是:
```

```
weburl: "/msdownload", visit counts: 10836
weburl: "/ie", visit counts: 9383
weburl: "/search", visit counts: 8463
weburl: "/isapi", visit counts: 5330
weburl: "/products", visit counts: 5108
set min support count as 291
```



分析

从上述图可以看出，大多数网站访问量都很小，不具有分析价值，访问量在第50位的地方作为边界值。

In [64]:

```
def get_itemset_support(itemset):
    ans = 0
    for transaction in transactions:
        if len(itemset) > len(transaction):
            continue
        exist = True
        for item in itemset:
            if item not in transaction:
                exist = False
                break
        if exist:
            ans += 1
    return ans



def Apriori(table, items, min_sup):
    new_table = []
    itemset_set = set()
    for itemset, _ in table:
```

```
        itemset_set.add(','.join(itemset))
    for itemset, _ in tqdm(table,
total=len(table)):
        for item in items:
            if item > itemset[-1]:  # 新增加的item
                mark = True
                for i in range(len(itemset)):  #
n-1 itemset
                    itemset_prior = itemset[:i] +
itemset[i+1:] + [item]
                    if ','.join(itemset_prior)
not in itemset_set:
                        mark = False
                        break
                if not mark:
                    continue
                new_itemset = itemset[:] + [item]
                new_support =
get_itemset_support(new_itemset)
                if new_support > min_sup:

new_table.append([new_itemset, new_support])
    return new_table


MIN_SUP = 500
items = [webid for webid, cnt in
```

```
webid2cnt.items() if cnt >= MIN_SUP]   # 初始项目集
table = [[[webid], cnt] for webid, cnt in
webid2cnt.items() if cnt >= MIN_SUP]   # 初始候选集

item_length = 1
while len(table) > 0:
    print(f"项目集长度: {item_length}, 项目集数量:
{len(table)}")
    new_table = Apriori(table, items, MIN_SUP)
    if len(new_table) > 0:
        table = new_table
    else:
        break
    item_length += 1
```

```
项目集长度: 1, 项目集数量: 35
100%|██████████| 35/35 [00:05<00:00,  6.89it/s]
项目集长度: 2, 项目集数量: 49
100%|██████████| 49/49 [00:00<00:00, 77.44it/s]
项目集长度: 3, 项目集数量: 28
100%|██████████| 28/28 [00:00<00:00, 728.34it/s]
项目集长度: 4, 项目集数量: 2
100%|██████████| 2/2 [00:00<00:00, 18935.91it/s]
```

In [67]:

```
# 计算置信度
def confidence(itemset):
    support_whole = get_itemset_support(itemset)
    support_prior =
get_itemset_support(itemset[:-1])
    return support_whole / support_prior

for itemset, cnt in table:
```

```
    support_rate = get_itemset_support(itemset) /
len(transactions)
    conf = confidence(itemset)
    print(f"项目集: {itemset}, 支持度:
{support_rate:.3f}, 置信度: {conf:.3f} 提升度:
{conf / support_rate:.3f}")
    # print(conf)
```

```
项目集: ['1001', '1003', '1018', '1035'], 支持度: 0.015, 置信度: 0.462 提升度: 30.093
项目集: ['1008', '1009', '1018', '1035'], 支持度: 0.020, 置信度: 0.673 提升度: 33.041
```

分析

从上述结果来看，最终确定了4个强关联的url,通过置信度和提升度可以看出，所选择的关联规则确实有用。

In [70]:

```
print("推荐的项目为: ")
for itemset, support in table:
    urls = [webid2weburl[item] for item in
itemset]
    print(urls)
```

```
推荐的项目为:
['"/support"', '"/kb"', '"/isapi"', '"/windowssupport"']
['"/msdownload"', '"/windows"', '"/isapi"', '"/windowssupport"']
```

结论

根据上述分析，推荐 support kb ispai windowssupport 作为一个导航栏菜单 或者 msdownload, windows, isapi, windowssupport 作为一个导航栏菜单