

实验四报告

程序语言: Python

姓名: 郑子睿

学号: 22920212204317

专业: 计算机科学与技术

内容: 在线广告

一、问题描述

问题表述:

有 N 个关键字竞价者, 每个竞价者指定了一个最大预算 b_i .
 Q 是一个关键字的集合。

每个竞价者 i 对一个关键字 $q \in Q$, 指定一个出价 C_{iq} .

竞价开始后, 关键字序列 $q_1, q_2, \dots, q_M (q_j \in Q)$ 实时到达, 每个 q_j 必须实时分配给某个竞价者 i 的广告以赚取收益 C_{iq_j} .

问题的目标是: 在满足竞价者对关键字匹配要求的基础上, 使总收益最大。

二、算法思想

本实验是对一篇论文中的方法的复现

[论文下载地址](#)

算法描述如下:

```
1  最新到来的关键字, 应该匹配给 $v$ 值最大的那个广告
2  其中:
3       $v = c(i) * CTR * ph(T(i))$ 
4      其中:
5       $c(i)$  = 该广告为该关键字 $i$ 的出价
6       $CTR$ : 该广告历史点击量 / 该广告历史显示总次数
7       $ph(x) = 1 - \exp(x - 1)$ 
8       $T(i) = m_i / b_i$ 
9      其中:
10      $m_i$ : 目前为止该广告的竞价者当日已经实际用掉多少钱
11      $b_i$ : 该广告的竞价者当日总预算
```

三、描述算法

函数传入两个参数:

1. `bidders`

储存了所有竞价者的:

- 姓名
- 每个关键字的报价
- 当日总预算
- 当日已使用的钱数
- CTR

2. adwords

随机生成的关键字序列

```

1 def calculateGrossIncome(bidders, adwords):
2     grossIncome = 0 # 总收益
3     for adword in adwords:
4         pqueue = queue.PriorityQueue() # 定义一个优先队列，便于查找最大的V
5         for name, info in bidders.items():
6             # 用公式算出每个竞价者的V，存入优先队列中
7             v = info[adword] * info['CTR'] * (1 - math.exp(info['当日已使用金
额'] / info['当日总预算'] - 1))
8             pqueue.put((-v, (name, info[adword])))
9             winner = pqueue.get()
10            print(f'关键字 "{adword}" 分配给了{winner[1][0]}，收益{winner[1][1]}
元')
11            grossIncome += winner[1][1]
12            bidders[winner[1][0]]['当日已使用金额'] -= winner[1][1]
13    return grossIncome

```

四、验证算法

具体代码

```

1 # 导入模块
2 import openpyxl as xlsx
3 import queue
4 import random
5 import math
6
7 # 读取excel中的数据
8 def load():
9     book = xlsx.load_workbook('data.xlsx')
10    biddersSheet = book['竞价者信息表']
11    # 创建bidderDict，用于储存每个竞价者的信息和出价
12    bidders = {}
13    for bidder in biddersSheet[2:biddersSheet.max_row]:
14        name = ''
15        for col in bidder:
16            header = biddersSheet.cell(1, col.col_idx).value
17            if header == '竞价者':
18                bidders[col.value] = {}
19                name = col.value
20            else:
21                bidders[name][header] = col.value
22    bidSheet = book['关键词出价表']
23    adwords = set([])
24    for bid in bidSheet[2:bidSheet.max_row]:
25        name = ''
26        for col in bid:

```

```

27         header = bidSheet.cell(1, col.col_idx).value
28         if header == '竞价者':
29             name = col.value
30         else:
31             adwords.add(header)
32             bidders[name][header] = col.value
33     return bidders, list(adwords)
34
35     # 计算总收益
36     def calculateGrossIncome(bidders, adwords):
37         grossIncome = 0
38         for adword in adwords:
39             pqueue = queue.PriorityQueue()
40             for name, info in bidders.items():
41                 v = info[adword] * info['CTR'] * (1 - math.exp(info['当日已使用金
42 额'] / info['当日总预算'] - 1))
43                 pqueue.put((-v, (name, info[adword])))
44             winner = pqueue.get()
45             print(f'关键字 "{adword}" 分配给了{winner[1][0]}, 收益{winner[1][1]}
46 元')
47             grossIncome += winner[1][1]
48             bidders[winner[1][0]]['当日已使用金额'] -= winner[1][1]
49         return grossIncome
50
51     # 主函数
52     if __name__ == '__main__':
53         data, adwords = load()
54         random.shuffle(adwords)
55         grossIncome = calculateGrossIncome(data, adwords)
56         print(f'最大总收益{grossIncome}元')

```

结果展示

竞价者信息表

	A	B	C	D	E
1	竞价者	当日已使用金额	当日总预算	CTR	V
2	A	10.00	100.00	50.00%	
3	B	30.00	100.00	50.00%	
4	C	10.00	60.00	20.00%	
5	D	66.00	100.00	50.00%	
6	E	10.00	100.00	20.00%	
7	F	0.10	1,000.00	50.00%	
8	G	2.00	100.00	50.00%	
9	H	80.00	300.00	40.00%	

关键词出价表

	A	B	C	D	E
1	竞价者	手机	照相机	笔记本电脑	平板
2	A	1.00	1.50	1.00	1.00
3	B	1.00	0.10	1.00	3.00
4	C	1.05	2.00	1.00	0.99
5	D	1.00	1.00	1.00	0.10
6	E	1.50	1.00	1.00	1.00
7	F	0.10	1.00	0.10	1.00
8	G	2.00	0.50	1.00	0.10
9	H	1.00	0.10	1.00	2.00

代码运行结果

```
关键字 "笔记本电脑" 分配给了G，收益1元
关键字 "手机" 分配给了G，收益2元
关键字 "平板" 分配给了B，收益3元
关键字 "照相机" 分配给了A，收益1.5元
最大总收益7.5元
```

五、结论

本文参考了现有论文的做法，做了代码的复现。

改进措施：可以结合离线算法