Web 信息处理与应用：课后作业 2
网页查询 + 排序 + 结果评估部分
请于 2020 年 11 月 19 日前将作业电子版发送至课程邮箱：ustcweb2019@163.com
作业文件与邮件标题命名：　　（姓名）_HW2

骆霄龙_PB18151853

1　计算题

1.1 给定以下词项的 idf 值，以及在三篇文档中的 tf，已知总文档数为 811,400，请完成如下计算任务：

	df	tf@Doc1	tf@Doc2	tf@Doc3
Car	**18,871**	**34**	**8**	**32**
Auto	**3,597**	**3**	**24**	**0**
Insurance	**19,167**	**0**	**51**	**6**
Best	**40,014**	**18**	**0**	**13**

1)　计算所有词项的 tf-idf 值。

过程：

```r
  {r}
N = 811400
car_df = 18871
auto_df = 3597
insurance_df = 19167
best_df = 40014
car_tf = c(34,8,32)
auto_tf = c(3,24,0)
insurance_tf = c(0,51,6)
best_tf = c(18,0,13)

idf_val <- function(x) {log10(N/x)}
tf_idf <- function(x) {
  if(x == 0) {
      return(0)
  }
  else {
    tf_idf = log10(x) +1
  }
}
```

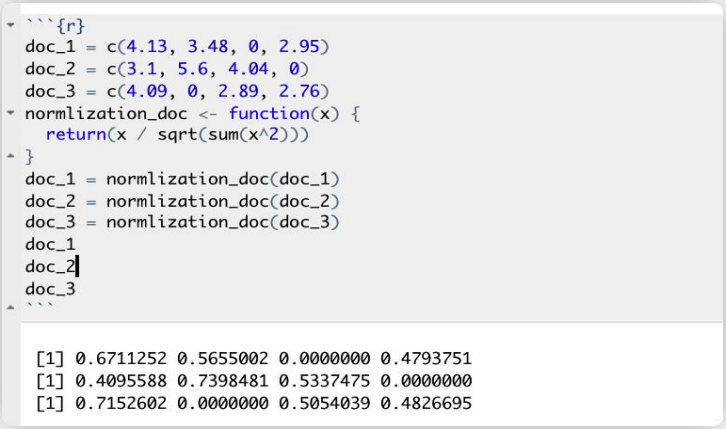
```r
```{r}

df = c(car_df,auto_df,insurance_df,best_df)
df = idf_val(df)
car_tf_1 = sapply(car_tf, tf_idf)
auto_tf_1 = sapply(auto_tf, tf_idf)
insurance_tf_1 = sapply(insurance_tf, tf_idf)
best_tf_1 =sapply(best_tf, tf_idf)
cat("car_tf-idf:      ", df[1] * car_tf_1, "\n")
cat("auto_tf-idf:     ", df[2] * auto_tf_1, "\n")
cat("insurance_tf-idf:", df[3] * insurance_tf_1, "\n")
cat("best_tf-idf:     ", df[4] * best_tf_1, "\n")
```

car_tf-idf: 4.135019 3.108583 4.092012
auto_tf-idf: 3.476101 5.601338 0
insurance_tf-idf: 0 4.404353 2.892485
best_tf-idf: 2.947693 0 2.762973
```

	Tf-idf@Doc1	Tf-idf@Doc2	Tf-idf @Doc3
Car	**4.13**	**3.10**	**4.09**
Auto	3.48	5.60	**0**
Insurance	**0**	**4.04**	**2.89**
Best	**2.95**	**0**	**2.76**

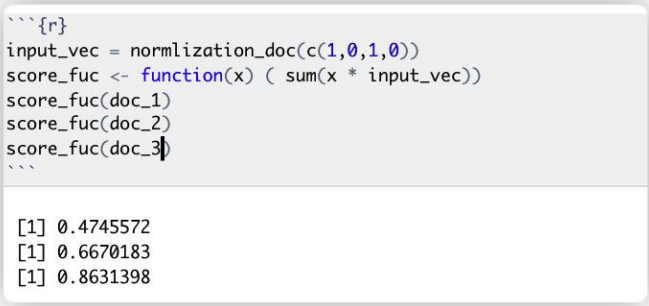
2) 试采用欧式归一化方法（即向量各元素平方和为 1），得到处理后的各文档向量化表示，其中每个向量为 4 维，每一维对应 1 个词项。



```r
doc_1 = c(4.13, 3.48, 0, 2.95)
doc_2 = c(3.1, 5.6, 4.04, 0)
doc_3 = c(4.09, 0, 2.89, 2.76)
normlization_doc <- function(x) {
  return(x / sqrt(sum(x^2)))
}
doc_1 = normlization_doc(doc_1)
doc_2 = normlization_doc(doc_2)
doc_3 = normlization_doc(doc_3)
doc_1
doc_2
doc_3
```

```
[1] 0.6711252 0.5655002 0.0000000 0.4793751
[1] 0.4095588 0.7398481 0.5337475 0.0000000
[1] 0.7152602 0.0000000 0.5054039 0.4826695
```

3) 基于 2)中得到的向量化表示，对于查询“car insurance”，计算 3 篇文档的得分并进行排序。其中，查询中出现的词项权重为 1，否则为 0。



```r
input_vec = normlization_doc(c(1,0,1,0))
score_fuc <- function(x) ( sum(x * input_vec))
score_fuc(doc_1)
score_fuc(doc_2)
score_fuc(doc_3)
```

```
[1] 0.4745572
[1] 0.6670183
[1] 0.8631398
```

所以排序结果为 Doc3 > Doc2 > Doc1 , 得分如上图所示。

1.2 考虑如下邻接矩阵表示的图（1 代表出边，0 表示无连接）

	节点 0	节点 1	节点 2	节点 3	节点 4	节点 5	节点 6
节点 0	0	0	1	0	0	0	0
节点 1	0	1	1	0	0	0	0
节点 2	1	0	1	1	0	0	0
节点 3	0	0	0	1	1	0	0
节点 4	0	0	0	0	0	0	1
节点 5	0	0	0	0	0	1	1
节点 6	0	0	0	1	1	0	1

1) 当 Restart 部分的随机跳转概率为 0.15 时，写出 PageRank 的（随机）转移概率矩阵。

```r
alpha = 0.15
n = 7
raw_data = rep(0,49)
p = matrix(data = raw_data, nrow = 7)
p[1,3] =1; p[2,c(2,3)] =1; p[3,c(1,3,4)] =1; p[4,c(4,5)] =1
p[5,c(7)] =1; p[6,c(6,7)] =1; p[7,c(4,5,7)] =1
for( i in 1:n) {
  p[i,] = p[i,] / sum(p[i,])
}
p =p* (1- alpha) + alpha / n
p
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
[1,] 0.02142857 0.02142857 0.87142857 0.02142857 0.02142857 0.02142857 0.02142857
[2,] 0.02142857 0.44642857 0.44642857 0.02142857 0.02142857 0.02142857 0.02142857
[3,] 0.30476190 0.02142857 0.30476190 0.30476190 0.02142857 0.02142857 0.02142857
[4,] 0.02142857 0.02142857 0.02142857 0.44642857 0.44642857 0.02142857 0.02142857
[5,] 0.02142857 0.02142857 0.02142857 0.02142857 0.02142857 0.02142857 0.87142857
[6,] 0.02142857 0.02142857 0.02142857 0.02142857 0.02142857 0.44642857 0.44642857
[7,] 0.02142857 0.02142857 0.02142857 0.30476190 0.30476190 0.02142857 0.30476190
```

2) 计算该矩阵所对应的 PageRank 向量（每一维表示一个节点的 PageRank 值）。

下图中 x 为初始 值， 经过 22 次迭代后收敛（eps = 1e-5），各个节点 pagerank 如下图

```r
eps = 1e-5
x = rep(1/7,7)
p_trans = t(p)
y = p_trans %*% x
iter = 1
while (max(abs(y-x))> eps) {
  x = y
  y = p_trans %*% x
  iter = iter + 1
}
cat("iteration times:    ", iter)
y
sum(y)
```

```
iteration times:     22
[1,] 0.05447258
[2,] 0.03726708
[3,] 0.11661631
[4,] 0.24312916
[5,] 0.21008516
[6,] 0.03726708
[7,] 0.30116262
[1] 1
```

此处1-7为pagerank

= sum(y)

3) 将邻接矩阵中“节点 2 指向节点 3”和“节点 6 指向节点 3”的两条边权重设为 0，其它不变。请计算该矩阵所对应的 Hub 值和 Authority 值向量。

```r
eps = 1e-4
raw_data = rep(0,49)
p = matrix(data = raw_data, nrow = 7)
p[1,3] =1; p[2,c(2,3)] =1; p[3,c(1,3)] =1; p[4,c(4,5)] =1
p[5,c(7)] =1; p[6,c(6,7)] =1; p[7,c(5,7)] =1
p_trans = t(p)
a0 = rep(1,7)
h0 = rep(1,7)

a1 = p_trans %*% h0
h1 = p %*% a1
k = 1
while(max(abs(a1-a0))> eps || max(abs(h1-h0))> eps) {
  a0 = normlization_doc(a1)
  a1 = normlization_doc(p_trans %*% h1)
  h0 = normlization_doc(h1)
  h1 = normlization_doc(p %*% a1)
  k = k + 1
}
a1  # Authority
h1  # Hub
k  # iteration timesa
```

```
          [,1]
[1,] 0.0005757488
[2,] 0.0005757488
[3,] 0.0015729749
[4,] 0.1684576058
[5,] 0.4980104117
[6,] 0.2725701319
[7,] 0.8057977729
```

Authority

```
          [,1]
[1,] 0.0007908197
[2,] 0.0010802799
[3,] 0.0010802799
[4,] 0.3350695846
[5,] 0.4051182022
[6,] 0.5421539765
[7,] 0.6554950206
```

Hub

```
[1] 107
```

iteration times

1.3 在由 **10,000** 篇文档构成的文档集中，某个查询的相关文档总数为 **10**，下面给出了针对该查询的前 **20** 个有序结果，其中 **R** 表示相关，**N** 表示不相关。

RRNNR NNNRN RNNNR NNNNR

请计算：

a) 该查询的 **P@10** 和 **P@20** 分别是多少？

- P@10 = 4/10 = 0.4
- P@20 = 7/20 = 0.35

b) 该查询前 **10** 篇文档和前 **20** 篇文档的 **F1** 值分别是多少？

- R@10 = 4/10，所以 F1@10 = 2PR/(P+R) = 0.373
- R@20 = 7/10，所以 F1@20 = 2PR/(P+R) = 0.467

c) 当该算法只返回前 **20** 个结果是，其简易 **AP** 值为多少？ AP: Average Precision

- AP = (1 +1 +3/5 + 4/9 + 5/11 + 6/15 + 7/20) / 7 = 0.607

假定该算法返回了全部 **10,000** 篇文档，上述 **20** 篇文档只是最开始的 **20** 个结果，那么

d) 该算法可能的最大 **AP** 值是多少？

- APmax = (1 +1 +3/5 + 4/9 + 5/11 + 6/15 + 7/20 + 8/21 + 9/22 + 10/23) /10 = 0.547

e) 该算法可能的最小 **AP** 值是多少？

- APmin = (1 +1 +3/5 + 4/9 + 5/11 + 6/15 + 7/20 + 8/9998 + 9/9999 + 10/10000) /10 = 0.425

2 问答题（言之有理即可）

2.1 请简述解决以下问题的思路：

a) 如何从多源情境信息（如手机的多种传感器信息）中，抽象出用户当前所处的状态或行为模式？

- 如可以以时间或者地点为标志，再结合一些其他传感信息如步频，周围环境温度等推断用户当前行为模式。

b) 在上述过程中，如何既体现用户的个性化因素，又减少用户个人记录稀疏的负面影响？

- 考虑结合用户的以往行为记录和用户的特点 将用户划分为几个大类，使得一个大类的用户具有一定的行为共性，再结合当前用户具体的信息在每个大类中用特有的规则进行推断。

2.2 用户在浏览网页时，可能通过点击“后退”按钮回到上一次浏览的页面。用户的这种回退行为（包括连续回退行为）能否用马尔科夫链进行建模？为什么？

- 不能，因为回退行为回到上一个（多个）页面是与上一个（多个）页面相关的，因此回退行为不具有马尔可夫性（下一个位置只与当前位置有关）

2.3 如何在网页排序的同时提升结果的多样化水平？如何在实现这一目的的同时保障算法的效率？

- 可以通过划分主题来实现多样化，即返回部分分主题排序的结果，同时可以结合用户的特性做一些相关性推荐来提升排序结果的多样化水平。
- 直接把提升多样化水平作为一个优化任务，再设计算法时就考虑保障多样化的反馈。