Web信息处理与运用实验三报告

- PB19051035 周佳豪
- PB18071496 李昱祁

实验完成说明

没有做跨域推荐，仅使用`DoubanMusic.txt`的数据。

预测的最终结果为:

提交时间	文件名称	Hit@20	Hit@100	NDCG@20	NDCG@100
2022-01-22 13:22:43	175_1642828963_result.txt	0.059409	0.243951	0.016103	0.049040

实验环境

`python 3.9`

运行程序说明

- `data`目录下存放着实验的原始数据，本次实验只用到了`DoubanMusic.txt`
- `src`目录下存放着实验源代码，分别为`dataset.py`,`train.py`,`sort.py`,`predict.py`
- 实验过程中会在`result`目录下产生`similarity.txt`与`sort_similarity.txt`两个中间文件以及最后的预测结果文件`result.txt`
- 如何开始实验
  - 首先，先运行`\src\train.py`，该程序会产生中间文件`similarity.txt`,大小为2G，大概需要4h。
  - 然后，再运行`\src\sort.py`,该程序会对上一步产生的中间文件`similarity.txt`内容进行排序，生成第二个中间文件`sort_similarity.txt`，大小为2G，该过程大概需要10min
  - 最后，运行`\src\predict.py`，该程序会读取`sort_similarity.txt`，然后生成最终的预测文件`result.txt`

实验思路

- 本次实验采用用户-用户协同过滤算法，忽略用户对音乐的打分，只考虑是否听过。
- 使用两个用户所听音乐`交集的大小/并集的大小`代表两个用户的相似度
- 按照相似度的大小为每个用户进行排序，然后基于相似度优先推荐相似度高的用户已听过的音乐推荐给该用户，推荐过程中要注意避免重复推荐

实验过程

- `\src\dataset.py`中设置了读取`DoubanMusic.txt`的程序，使用字典储存数据，键为用户id，值为用户所听过音乐组成的数组

```
class MusicDataset(torch.utils.data.Dataset):
    def __init__(self,datafile:str)-> None:
        super(MusicDataset,self).__init__()
        self.id_Music = {}
        with open(datafile,"r") as fr:
            for line in fr.readlines():
```

```
                if(line=='\n'):
                    break
                index = line.strip().split('\t')[0]
               # print('index=',index)
                self.id_Music[int(index)]=[]
                for s in line.strip().split('\t')[1:]:
                    #print('s=',s)
                    #print(float(s))
                    t = s.split(',')[0]
                    #print('t=',t)
                    self.id_Music[int(index)].append(int(t))
                    #print(self.id_Music[int(index)])
        for key in self.id_Music:
            self.id_Music[key] = np.array(self.id_Music[key])
```

- `\src\train.py`通过引用`dataset.py`中的`MusicDataset`类读取数据文件，然后计算每两个用户之间的相似度，最后将结果储存在`\result\similarity.txt`,其中文件的每一行为用户（设为甲）与其他所有用户（包括自己）的相似度，用`\t`分隔。第一个元素是用户甲ID，之后是甲与其他所有用户的相似度，其中用户ID、相似度两个用`,`区分开。

```
from dataset import MusicDataset
import numpy as np

outpath = "./result/similarity.txt"
data = MusicDataset("./data/DoubanMusic.txt")

def similarity(a,b):
    intersection = np.intersect1d(a,b)
    union = np.union1d(a,b)
    result = intersection.size / union.size
    return result

#compute the similarity each id
#result = {}
with open(outpath,'w') as fw:
    for id in data.id_Music.keys():
        #store the similarity between id and others besides id itself
        print("id: ", id)
        fw.write(str(id)+'\t')
        #result[id]=[]
        for key in data.id_Music.keys():
            #print("key: ", key)
            similar = similarity(data.id_Music[id],data.id_Music[key])
            #result[id].append(float(similar))
            if(similar > 0):
                fw.write(str(key)+','+str(format(similar,'.4f'))+' ')
        fw.write('\n')
```

- `\src\sort.py`对`\result\similarity.txt`进行排序，对每一行将相似度高的用户排在前面（显然自己与自己相似度最高），最后将排序后的结果储存在`\result\sort_similarity.txt`中

```
import functools
import numpy as np

inpath = "./result/similarity.txt"
outpath = "./result/sort_similarity.txt"
```

```
def custom_sort(x,y):
    a = float(x.strip().split(',')[1])
    b = float(y.strip().split(',')[1])
    if a > b:
        return -1
    if a < b:
        return 1
    return 0

with open(inpath, 'r') as fr,open(outpath, 'w') as fw:
    for line in fr.readlines():
        id = line.strip().split('\t')[0]
        print('id=', id)
        fw.write(str(id)+'\t')
        score_list = line.strip().split('\t')[1].split(' ')
        #print(score_list)
        x = sorted(score_list,key=functools.cmp_to_key(custom_sort))
        for val in x:
            fw.write(val+' ')
        fw.write('\n')
```

- `\src\predict.py`，根据`\result\sort_similarity.txt`中的相似关系，优先将相似度高的用户听过的音乐推荐给该用户，直至推荐了100个音乐。代码使用有序集合`Ordered-set`来避免重复推荐。

```
import numpy as np
from dataset import MusicDataset
from ordered_set import OrderedSet

path = "./result/sort_similarity.txt"
data = MusicDataset("./data/DoubanMusic.txt")


with open(path, 'r') as fr,open("./result/result.txt",'w')as fw:
    for line in fr.readlines():
        s = line.strip().split('\t')
        index = int(s[0])
        print('index=',index)
        fw.write(str(index)+'\t')
        a = OrderedSet()
        for value in s[1].strip().split(' '):
            if(len(a)>=100):
                break
            target = int(value.strip().split(',')[0])
            for j in data.id_Music[target]:
                if(len(a)<=99):
                    a.add(str(j))
                else:
                    break
        count=0
        for i in a:
            if(count<99):
                fw.write(i+',')
                count = count +1
            else:
                fw.write(i)
```

```
                count = count +1
        fw.write('\n')
```

实验结果

提交时间	文件名称	Hit@20	Hit@100	NDCG@20	NDCG@100
2022-01-22 13:22:43	175_1642828963_result.txt	0.059409	0.243951	0.016103	0.049040

实验总结与反思

- 用户所听的音乐矩阵是稀疏的，一些小众项目无法被推荐
- 本次实验复杂度过高，计算相似性的时间复杂度为$O(n^2)$
- 推荐过程中，没有考虑用户的评分，所有可能会推荐一些“难听”的音乐。