数据挖掘实验报告

1. VSM

1.1 预处理

Input ： 单篇文件
Processing：去除数字，标点，词缀，等，进行分词，生成处理过的词表
Output：该文件生成的 wordList

主要实现：
C:\Users\eric\Desktop\VSM\Vsm\preProcessing.py

```
def DealwithFile(filename):
    #fw = codecs.open(filename, 'r', )
    f = open(filename,'r',errors='ignore')
    text = f.read()
    #去除读取文件后的格式（换行 缩进）
    text = text.replace('\r',' ').replace('\n',' ').replace('\t',' ')
    remove_digits = str.maketrans('', '', string.digits)
    text = text.translate(remove_digits)
    f.close()
    return text
#(2)去除标点和大写,数字，分词
def remove_tokens(text):
    lowers = text.lower()  # 大小写
    # 去除标点符号
    remove_punctuation_map = dict((ord(char), None) for char in string.punctuation)
    no_punctuation = lowers.translate(remove_punctuation_map)
    tokens = nltk.word_tokenize(no_punctuation)
    return tokens

#（2）提取词干
def Stemming(wordlist):
    LancasterStem = nltk.LancasterStemmer()
```

1.2 生成词典

Input: 生成的 wordList
Processing：构造词典的数据结构
{“key1”：32}//word 和其全局的频率
Output: Dict in redis
目的：筛选 keyword（选择频率高于 15 的词，将词典从 80000 项目讲到 8703 项）

```python
def dealwithfile(path, Redis):
    # 把每个文件生成的词典写入redis，注意redis是动态更新的
    # 处理文件生成wordlist
        wordlist=PreProcessing(path)
        #print(wordlist)
        # redis插入和赋值
        for word in wordlist:
            if word != "" and Redis.exists(word):
                num = int(Redis.get(word))
                Redis.set(word, str(num + 1))
            elif word != "" and word != " ":
                Redis.set(word, 1)
def Read_all_files(rootdir):
    list = os.listdir(rootdir)  # 列出文件夹下所有的目录与文件
    for i in range(0, len(list)):
        path = os.path.join(rootdir, list[i])
        if os.path.isdir(path):
           Read_all_files(path)
        elif os.path.isfile(path):
            dealwithfile(path, Redis)


def mainfuc():
    # 以文件夹为单位读
```

1.3 TF_IDF 计算

Input:词典（主要时确定需要保留那些 word）
Processing：遍历文章进行词频统计，意在形成下述的数据结构

```
{
   'word1': //keyword
            {
                'file1': 23, //在 file 中出现的次数
                'file2': 21,
                。。。。。。
            }
}
```

Output :TF_IDF Dict

在统计的同时生成 TF*IDF

```python
def CalcIDF(word,Redis):
    #1.这个词总的出现次数
    count=len(json.loads(Redis.get(word)))
    #2.total
    total=18828
    return math.log(total/(count+1))

def UpDateRedisToTF_IDF(Redis):
    keys=Redis.keys()
    for key in keys:
        IDF = CalcIDF(key,Redis)
        pathlist=json.loads(Redis.get(key))
        for path in pathlist:
            TF=pathlist[path]
            pathlist[path]=TF * IDF
        Redis.set(key,json.dumps(pathlist))
        print('1wf')

def DwonToFile(Redis,localdir):
    f=open(localdir,'w',errors='ignore')
    keys=Redis.keys()
    Dict={}
    for key in keys:
        Dict[key]=Redis.get(key)
    f.write(str(Dict))
```

```
C:\WINDOWS\system32\cmd.exe - redis-cli  -a 123456
8788) "sack"
8789) "bgrubbdantenmsuedu"
8790) "assocy"
8791) "syquest"
8792) "comp.windows.x67515"
8793) "bash"
8794) "faceoff"
8795) "enh"
8796) "demograph"
8797) "staffordvaxwinonamsusedu"
8798) "triangl"
8799) "12315sci.electronics"
8800) "abound"
8801) "fetch"
8802) "munny"
8803) "plat"
8804) "res"
8805) "tick"
8806) "wors"
8807) "nearby"
8808) "aafreenetcarletonc"
8809) "resta"
8810) "jfcathenamitedu"
8811) "darn"
8812) "ihl"
8813) "mithra"
8814) "xb"
8815) "unclear"
8816) "interv"
8817) "amp"
8818) "alon"
8819) "rgs"
8820) "cul"
8821) "reconstruct"
8822) "khan"
8823) "hadescoosdartmouthedu"
8824) "peg"
8825) "raytrac"
8826) "anarch"
```

1.4 形成 VSM 数据

2. KNN

2.1 面向文件

先转换第一部分生成的 Dict 成为 File Version（之前工作出的纰漏，这里弥补，是因为没有想明白的问题，以后要先想明白再动手。）

```
def Convert(Redis):
    keys=Redis.keys()
    for key in keys :
        info=json.loads(Redis.get(key))
        #print(info)
        for path in info:
            pathkey=path
            pathkey=pathkey.rstrip(string.digits)
            pathkey = '12315' + pathkey
            if not Redis.exists(pathkey):

                Redis.set(pathkey,json.dumps({'huang':0}))
                print('2')
            else:
                WordInfo=json.loads(Redis.get(pathkey))
                s=''
                if key in WordInfo.keys():
                    s=WordInfo[key.decode()]
                    WordInfo[key.decode()]=info[path]+s
                else:
```

2.2 计算 KNN

根据已经形成的文件词典，以此计算 KNN，选取最大值，进行划分

```
for key in keys:
    if key[0:3] == '123':
        continue
    else:
        Redis.delete(key)
print('update finishing..')
files=Redis.keys()
cosList=[]
for file in files:
    cos=CalculateKNN(json.loads(Redis.get(file)),InputWordDict)
    cosList.append({file:cos})
print("KNN Processing finished")
cosList.sort()
print(cosList[0][0],"  :::  ",cosList[1][0],"--------- are likey to be the
```

2.3 命中率

选取了 10 个文件进行随机测试，正确划分完成了 6 次，准确率 60 左右，进一步的准确率还需要更多的样本进行支撑