人工智能实验报告

学院： 数据科学与计算机学院

专业： 计算机科学与技术

班级： 16 级教务 2 班

学期： 2018-2019 年秋季学期

学号： 16337327

姓名： 郑映雪

实验题目

TF-IDF 实验+KNN 实验

实验内容

算法原理

1、准备

首先对于每个句子我们得把它分成单词，形成不重复词汇表。在这一步可以使用 python 带的.split()函数，对于 TF-IDF 文件读取，使用制表符分割，对于另外两个 csv 文件读取，使用逗号分割。由于分割后的对象是列表形式，所以只要根据下标选择的相应想要的属性即可。

2、TF-IDF 原理

在本实验中，我们将所有文本作为一组向量，利用文本数据处理使其具有可计算性。将所有文本中的所有词划分，去除重复词语，形成不重复词汇表。

根据词汇在文本的词汇与否形成一个 0-1 矩阵——one-hot 矩阵。对此矩阵逐行进行归一化处理，即可得到 TF 矩阵。TF 矩阵归一化的作用是可以防止偏向某个文本过长的情况，即$\mathrm{TF_{I,j}} = \frac{n_{i,j}}{\sum n_{k,j}}$。

IDF 矩阵表示逆向文件频率，检测指标与不重复词汇表中每个单词有出现的文章数相关，公式为$\log(\frac{|D|}{1+\{j \in d_j\}})$，其中 D 为文章总数，j 为出现该次的文章个数。注意到分母中多了一个“+1”，是为了防止出现分母为 0 的情况。

于是我们得到 TF-IDF 矩阵，即$\mathrm{TFIDF_{i,j}} = tf_{i,j} * idf_i$。这个矩阵结合了两个矩阵的特点，倾向于高频率出现但重复率较低的词汇。

3、KNN 最近邻算法原理

KNN 最近邻是监督式学习的一个机器模型。通俗来说就是，如果一个样本与训练集中的 K 个样本最为相似，那么这个样本的值就等于这 K 个样本的值中的众数。作为对“相似度”的刻画，向量之间的距离就是一个很好的标准。通过矩阵构造得出每一个文本的向量，计算距离后找出最近的 K 个向量，值即为这 K 个向量中的众数。距离计算的选择有街区距离、欧氏距离、极大距离和余弦距离等。但在 K 取不同值时以及取不同的计算距离的方法时，分类结果可能会有显著不同，所以要拨出验证集进行验证，找到使准确率最高的 K 和最佳计算距离的方法。以上是进行使用 KNN 近邻算法对文本进行分类的方法。KNN 模型还可以用于回归。对于距离最近的 K 个训练文本，加权计算每一个测试文本对于不同分类的概率，通过相关系数来测试。

伪代码

1、TF-IDF 伪代码

①形成 TF 矩阵的伪代码如下所示：先在每一个文本中对不重复词汇表中的每一个单词计数，然后对每一行计算和，再在每一行的每个数除以该行的和进行归一化，这一步的矩阵运算在 python 里较方便实现。

```
for datas in data:
  split=datas.split(' ')
  for varis in vari:
    tmp.append(split.count(varis))#对每个文本中每个单词计数
  tf.append(tmp) #添加行以形成矩阵
sum=tf.sum(line)
tf=tf/sum
```

②形成 IDF 矩阵的伪代码如下图所示：先对不重复词汇表中的每一列词判断是否在文章中，即形成 one-hot 矩阵，再代入公式进行计算。

```
for varis in vari:
  i=0
  for datas in data:
    tmp=datas.split(' ')
    if varis in tmp:
```

```
            i+=1
        idf.append(log(len(data)/(1+i),e))
```

③逐列进行相乘，得到最终的 TFIDF 矩阵。

```
    for i in range(len(vari)):
            tf[:,i] *= idf[i]
    tfidf=tf
    return tfidf
```

2、KNN 实验伪代码

a.分类

①生成 one-hot 矩阵，此处伪代码与上面创建 IDF 矩阵的前半部分相同，故不赘述。

②两个 one-hot 矩阵进行计算算出距离。（此处采用欧氏距离计算）

```
    second=tile(va[i],(da.shape[0],1))-da #此处为每一行验证集复制扩充为矩阵后与训练集矩阵进行减运算
    second=second.getA() ** 2 #将矩阵转化为数组，逐一乘方
    second=mat(second) #又转为矩阵
    sumi=second.sum(axis=1)#行求和
    sumi=sumi.getA() ** 0.5 #开平方
```

③将距离排序。

```
sorted_x = sorted(distance)
```

④前 K 个元素即为最近元素，此处取 K 为 4。使用函数找到众数返回。

```
    for k in range(4):
      new.append(emotion[most[k])
    newemotion.append((new).most())
```

b.回归

①生成 one-hot 矩阵与求取距离、将距离排序的伪代码与上文相同，故不赘述。

②根据加权概率公式生成测试样本各个情感的概率。注意到其中有**距离为 0** 的情况，我处理这个情况的方法是加上一个非常小的数。各行的处理如下：

```
    for k2 in range(6):
      tmp2=0
```

```
for k1 in range(4):
  if distance[k1]==0:

    tmp2+=depro[distance.count()/(distance[k1]+0.0000001)
  else:
    tmp2+=depro[distance.count()/distance[k1]
tmpline.append(tmp2)
```

③为了让各个情感概率相加为 1，所以进行归一化处理后再返回测试样本的情感概率矩阵。

```
tmpsum=sum(tmpline)
tmpline /= tmpsum
vapro.append(tmpline.tolist())
```

关键代码截图

1、TF-IDF 实验

①生成不重复词汇表。

```
# 生成不重复的词汇表。data:测试集样本的列表
def tran(data):
    voca = []
    for sentence in data:
        tmp = sentence.split(' ') #逐行分离各个词语
        for tmp1 in tmp:
            voca.append(tmp1)
    voca2 = sorted(set(voca), key=voca.index)  # 利用set函数去除列表中重复元素
    return voca2 #返回不重复的词汇表
```

②生成 TF-IDF 矩阵

```
def tfidf(data, vari, num):
    #生成TF矩阵
    tf = []
    idf = []
    for datas in data:
        tmp=datas.split(' ')#对每行语句的词分开
        tmp2=[]
        for varis in vari:
            tmp2.append(tmp.count(varis))#计算分开的所有词里该词的个数，加入该行的列表中
        tf.append(tmp2)#列表添加行元素成为嵌套列表
    tf=mat(tf)#嵌套列表转化为矩阵
    sum=tf.sum(axis=1)#对矩阵进行各行求和
    tf=tf/sum#归一化处理

  #生成idf矩阵
  for varis in vari:
      i=0
      for datas in data:
          tmp=datas.split(' ')
          if varis in tmp:
              i+=1#判断词是否在语句的词组列表中，如果在则该词存在的文章数+1
      idf.append(math.log(len(data)/(1+i),math.e))#进行idf计算
  for i in range(len(vari)):
      tf[:,i] *= idf[i]#tf*idf
  tfidf=tf
  return tfidf #返回计算好的TF-IDF矩阵
```

2、KNN 分类实验

①one-hot 矩阵

```
#生成one-hot矩阵
def onehot(data, vari):
    res = []
    for datas in data:
        tmp = datas.split(' ')#将每个词语分割成词组
        tmp2=[]
        for varis in vari:
            if varis in tmp:#判断不重复词汇表里的词是否在样本中
                tmp2.append(1)
            else:
                tmp2.append(0)
        res.append(tmp2)
    res2=mat(res)
    return res2
```

②分类操作

```
#da：训练集的one-hot矩阵，va：验证集one-hot矩阵
def classification(da,va,emotion):
    newemotion=[]
    for i in range(va.shape[0]):
        tmp=tile(va[i],(da.shape[0],1))-da #将一行测试样本行复制扩展为矩阵，并将此矩阵与训练基矩阵做减运算
        tmp=tmp.getA() ** 2 #将矩阵转化为数组逐个乘方
        tmp=mat(tmp)
        sumi=tmp.sum(axis=1)#对做了减运算的矩阵进行每行都进行相加，得到一个每行和的矩阵
        sumi=sumi.getA() ** 0.5 #开方，即得到这条验证集语句与所有训练集语句的欧式距离

        distance={}
        for j in range(len(sumi)):
            distance[j]=sumi[j][0]
        sorted_x = sorted(distance.items(), key=operator.itemgetter(1))  # 这里按照value排序
        distance=sorted_x
        newemotio=[]
        for k in range(4):#取K=4
            newemotio.append(emotion[distance[k][0]])
        newemotion.append(Counter(newemotio).most_common(1)[0][0])#找众数
    return newemotion
```

3、KNN 回归实验

①one-hot 矩阵代码与分类中的 one-hot 代码相同，故不赘述。

②回归计算概率。此处计算距离的代码与回归中相同，故不赘述，只贴上计算概率的相关代码。

```
    tmpline=[]
    for k2 in range(6):#对每个测试集的各个情感概率进行计算
        tmp2=0
        for k1 in range(4):#取K=4
            if distance[k1][1]==0:#考虑到距离为0的情况，此处加上一个极小的数。
                tmp2+=depro[distance[k1][0]][k2]/(distance[k1][1]+0.00000000000000000001)
            else:
                tmp2+=depro[distance[k1][0]][k2]/distance[k1][1]
        tmpline.append(tmp2)
    tmpsum=sum(tmpline)
    tmpline /= tmpsum #在各行概率计算完毕后，进行归一化处理，以让各个情感概率为1
    vapro.append(tmpline.tolist())
return vapro#返回情感概率矩阵，以便进行相关度测试
```

创新点/优化

1、不同距离表达方式的选择

曼哈顿距离与欧式距离是最常用的，将代码微调，加入曼哈顿距离的选择，可以更好地通过验证集准确率的比较判断使用哪种距离准确度更高。

2、对于 K 值的测试过程

在分类中，在还未确定下来选择使用哪个 K 值以及哪个距离测量时，写个嵌套循环，在有限范围内（此处我取不超过训练集大小的一半）找到最优的 K 值和距离方式。由于对于固定的 K 和距离完全跑出一个准确度只需几秒，所以遍历方法是可行的（就比如我可以边写这个实验报告边等它跑完）。

在回归中，也可以这么判断，只不过指标由准确度改为了相关系数。

可以将代码改为如下形式：

```
for distancing in range(1,3):
    if distancing==1:
        print("The Manhattan Distance:\n")
    else:
        print("The Eulidean Distance:\n")
    for k in range(1,50):
        vapro=regression(onehotmatrix, vaonehotmatix,depro,k,distancing)
```

采用不同的 distancing 和 k 遍历，以找到最优的取值。

对于分类，评测准确度：

```
    accuracy=flag/len(testemotion)
    print('k=',k,'  accuracy:',float(accuracy))
    if acumax<accuracy:
        acumax=accuracy
        kmax=k
        proper=distancing
```

对于回归，评测相关系数：

```
for i in range(6):
    suml += np.corrcoef(vapro[i],vapro1[i])[0][1]
suml /=6
print("K=",k,"  Correlation:",suml)
if suml>summax:
    summax=suml
    kmax=k
    proper=distancing
```

具体的结果将在下文展示。

实验结果及分析

实验结果展示（优化前）

1、TF-IDF 矩阵展示

此处仅为部分展示，并且按照 TA 后来的要求删去了 0 值，仅保留非 0 值。

16337327_ZhengYingXue_TFIDF.txt

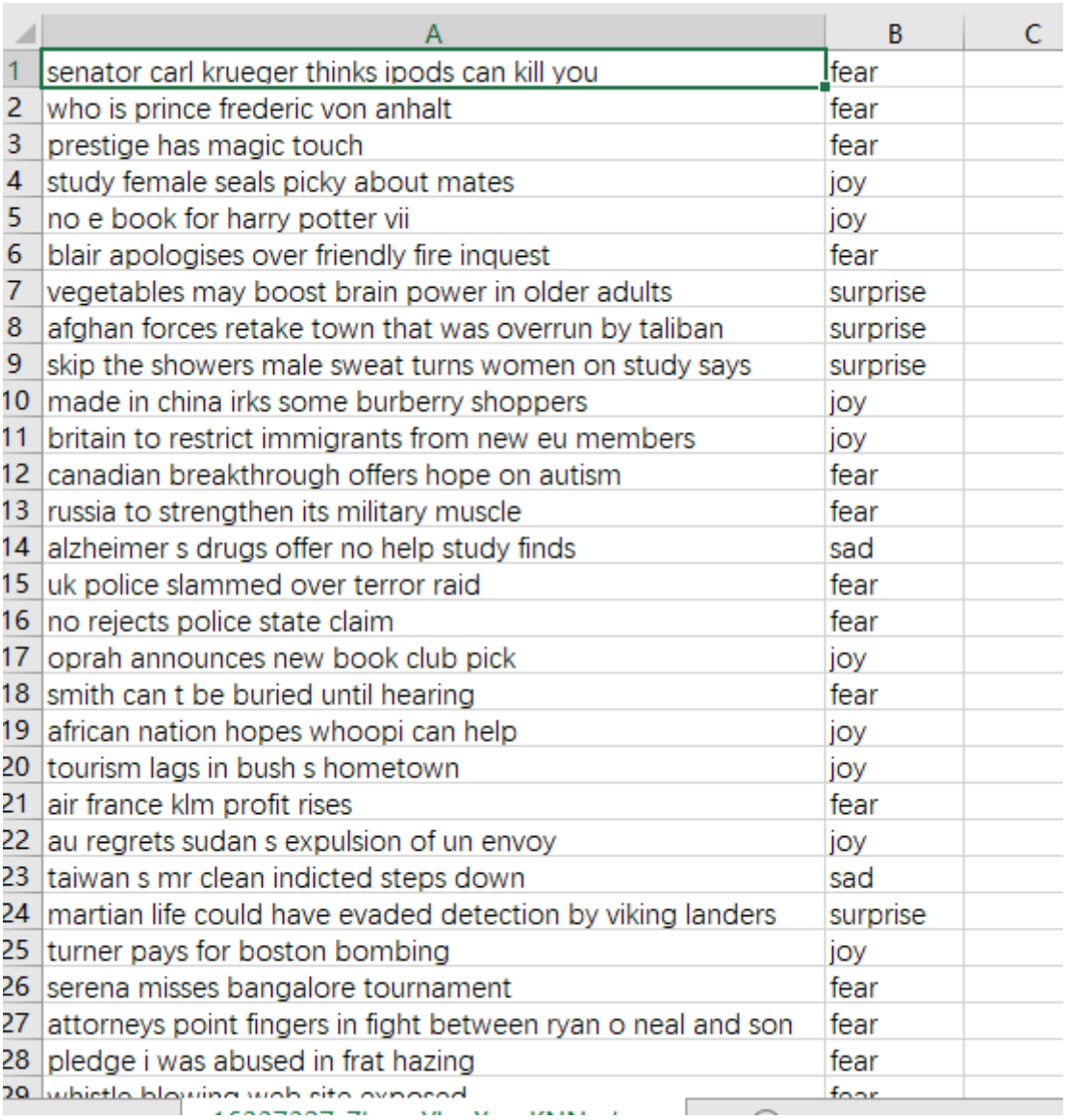
```
1.0724244197979087 1.0048469017798813 0.8217448536685299 0.5393122335394619 1.0048469017798813 0.7604573903143101
1.435349834556877 1.6086366296968633 0.5829757884376643 1.6086366296968633
0.7604573903143101 0.8893223716865571 1.0724244197979087 0.8413753596112603 0.5535052015961798 0.919709297818883
2.1448488395958174 1.190781879285995 1.9137997794091692
0.9568998897045846 0.672775207664847 1.0724244197979087 0.919709297818883 0.8041847677255587 1.0724244197979087
0.8893223716865571 1.0724244197979087 0.8893223716865571 0.4683675976351813 0.7604573903143101 1.0724244197979087
0.8302578023942697 1.6086366296968633 1.5072703526698221 1.6086366296968633
0.919709297818883 1.0724244197979087 0.7366072497075313 0.400050980105843 0.8636305917153475 0.9568998897045846
1.9137997794091692 2.1448488395958174 1.6434897073370598
0.6379332598030564 0.7149496131986058 0.6698979345199209 0.4591068250881563 0.21621223878392368 0.6698979345199209
0.35954148902630795 0.6379332598030564 0.5928815811243714 0.23230456632597438 0.374054842628204 0.7149496131986058
0.19459101490553132 0.20907410969337695 0.5181783550292085 0.6434546518787454 0.6029081410679289 0.5741399338227507
0.7604573903143101 0.29361961405425785 0.8041847677255587 0.6972091200301597 1.0724244197979087 0.7157467258818637
0.7604573903143101 1.0048469017798813 1.4962121232440464 1.0048469017798813 1.0724244197979087
0.5335934230119344 0.6434546518787454 0.6029081410679289 0.33210312095770 79 0.6434546518787454 0.3693706494862252 0
0.9125488683771723 0.8263922851586815 0.38918202981106265 1.1482798676455015 1.1482798676455015
0.17937716567312748 0.35098033399122014 0.4104564792399495 0.32165247645134915 0.25896104489644894 0.39859873463785
0.5703430427357327 0.2613426371167212 0.2202147105406934 0.5368100444113978 0.6477229437865106 0.5610795462165175 0
0.4683675976351813 0.8893223716865571 1.0048469017798813 1.0724244197979087 1.0724244197979087 1.0724244197979087
```

2、分类结果展示（此处在优化前暂定使用欧氏距离,K=4）

先在**验证集**上将结果输出为 csv 文件，部分截图如下所示：

	A	B
1	europe retain trophy with big win	fear
2	senate votes to revoke pensions	fear
3	the amounts you have to pay for a l	surprise
4	pair of satellites will document sun ir	fear
5	malaysian airasia x to fly in july	sad
6	dow hits new record eyes	joy
7	bathing mom awakes to find baby c	fear
8	we re a pretty kind bully	joy
9	women in their s are perfectly good	fear
10	hands on doomsday clock move for	fear
11	italy to hold no show trial of bronx c	joy
12	what the godfather of soul meant to	sad
13	snow begins heavy accumulations e	sad
14	millionaire secret santa dies	sad
15	martian life could have evaded dete	fear
16	gunman was srebrenica survivor	joy
17	world tourism sets record in	joy
18	earthlink ceo dies at age	joy
19	goal delight for sheva	fear
20	late penalty costs roma the points	fear
21	house of cards actor ian richardson	fear
22	even before its release world climate	joy
23	mln birds to be vaccinated from birc	joy
24	speaker hastert testifies before pane	fear
25	we lit our last candle yesterday	fear
26	putin vows to tackle illegal immigrat	fear
27	mideast summit edns with little proc	fear

在未优化前，对**测试集**的测试结果的部分截图如下：



	A	B	C
1	senator carl krueger thinks ipods can kill you	fear	
2	who is prince frederic von anhalt	fear	
3	prestige has magic touch	fear	
4	study female seals picky about mates	joy	
5	no e book for harry potter vii	joy	
6	blair apologises over friendly fire inquest	fear	
7	vegetables may boost brain power in older adults	surprise	
8	afghan forces retake town that was overrun by taliban	surprise	
9	skip the showers male sweat turns women on study says	surprise	
10	made in china irks some burberry shoppers	joy	
11	britain to restrict immigrants from new eu members	joy	
12	canadian breakthrough offers hope on autism	fear	
13	russia to strengthen its military muscle	fear	
14	alzheimer s drugs offer no help study finds	sad	
15	uk police slammed over terror raid	fear	
16	no rejects police state claim	fear	
17	oprah announces new book club pick	joy	
18	smith can t be buried until hearing	fear	
19	african nation hopes whoopi can help	joy	
20	tourism lags in bush s hometown	joy	
21	air france klm profit rises	fear	
22	au regrets sudan s expulsion of un envoy	joy	
23	taiwan s mr clean indicted steps down	sad	
24	martian life could have evaded detection by viking landers	surprise	
25	turner pays for boston bombing	joy	
26	serena misses bangalore tournament	fear	
27	attorneys point fingers in fight between ryan o neal and son	fear	
28	pledge i was abused in frat hazing	fear	
29	whistle blowing web site exposed	fear	

3、回归结果展示（在优化前此处暂取 K=4）

对于**验证集**，将得到的概率生成矩阵，部分截图如下所示，完整数据请查看文件。

	anger	digust	fear	joy	sad	surprise
0	0.193896	0.051839	0.232109	0.268167	0.145089	0.108899
1	0.111299	0.017859	0.352723	0.233564	0.187477	0.097079
2	0.172382	0	0.298272	0.25175	0.102104	0.175492
3	0.235482	0	0.33169	0.233006	0.113857	0.085966
4	0.096959	0.024774	0.323014	0.090299	0.319237	0.145717
5	0.0199	0.050001	0.19248	0.38081	0.037601	0.319208
6	0.111637	0.043226	0.382888	0.199421	0.1822	0.080627
7	0.042876	0.02239	0.209336	0.609719	0.066682	0.048997
8	0.170703	0.0109	0.266589	0.272522	0.082144	0.197143
9	0.073656	0.018082	0.354263	0.391518	0.06564	0.096842
10	0.031844	0.012607	0.239612	0.54606	0.036346	0.133531
11	0.026505	0.044209	0.24785	0.026505	0.557612	0.097319
12	0.04699	0.112453	0.239222	0.058282	0.352999	0.190055
13	0.025137	0.01537	0.211394	0.197964	0.393265	0.15687
14	0.122151	0.055382	0.347242	0.230647	0.214072	0.030507
15	0.025507	0	0.180074	0.454991	0.193895	0.145534
16	0.061073	0	0.26367	0.403747	0.147077	0.124432

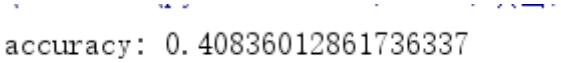
在未优化以前，对**测试集**的测试结果如下：

	anger	digust	fear	joy	sad	surprise
0	0.186509	0	0.281214	0.235165	0.217114	0.079998
1	0.236341	0	0.329273	0.234291	0.113655	0.08644
2	0.169311	0	0.216747	0.226513	0.291877	0.095552
3	0.020344	0	0.196766	0.58718	0.038438	0.157273
4	0.077052	0.052376	0.19248	0.40176	0.097127	0.179204
5	0.055271	0.093889	0.326286	0.057373	0.243616	0.223565
6	2.52E-22	0	0.0926	0.4444	5.39E-22	0.463
7	0.18666	0	0.260875	0.256888	0.138941	0.156636
8	0	0.098607	0	0.287759	0	0.613634
9	0.069902	0.0174	0.325308	0.354184	0.078902	0.154304
10	0.092556	0	0.277391	0.380551	0.158936	0.090566
11	0.109085	0	0.363544	0.282778	0.148695	0.095898
12	0.097242	0.021474	0.356498	0.23988	0.172974	0.111933
13	0.12165	0	0.243525	0.218425	0.357425	0.058975
14	0.078933	0.101528	0.477746	0.057374	0.129056	0.155363
15	0.224298	0.018481	0.364809	0.227174	0.11012	0.055118
16	0.081941	0.05834	0.25678	0.471018	0.098844	0.033077
17	0.1755	0.022386	0.201106	0.261885	0.083492	0.25563
18	0.032415	0	0.279884	0.439735	0.173182	0.074784
19	0.029472	0	0.208065	0.406919	0.224035	0.131508
20	0.245073	0.041104	0.251007	0.235176	0.164501	0.063139
21	0.030176	0	0.21303	0.450986	0.229381	0.076427
22	0.029311	0	0.247025	0.224523	0.355463	0.143678
23	0.117612	0	0.058806	2.77E-21	0.117612	0.705971

评测指标展示（优化前）

1、分类评测指标——准确度

在优化以前，简单在验证集上试了几个 K 值，其中最大的准确度出现在 K=4 时，如下所示：



但这个值是不是一定范围内最大的准确度尚且未可知，所以需要测试调参数。

2、回归评测指标——相关系数

此处在优化前仍然简单试了几个 K 值，同样当 K=4 时刚好由“加油哦小辣鸡”变为“低度相关 666”。但刚达 0.3 一定不是最大值，所以我也想通过调参数优化一下。

	anger	disgust	fear	joy	sad	surprise
r	0.205224476	0.216048543	0.346166928	0.399977026	0.320000441	0.317057678
average	0.300745849					
evaluation	低度相关 666					

实验结果展示（优化后）

1、优化过程

在一定范围内改变距离方式与 K 值，展示不同情况下的评测指标，从而选择最优的取法。

2、分类的优化过程展示

取 K 范围为 n/2。不同的评测指标如下：

首先，对于曼哈顿距离，结果如下图：

```
The Manhattan Distance:

k= 1   accuracy: 0.14790996784565916
k= 2   accuracy: 0.14790996784565916
k= 3   accuracy: 0.14790996784565916
k= 4   accuracy: 0.14790996784565916
k= 5   accuracy: 0.14790996784565916
k= 6   accuracy: 0.14790996784565916
k= 7   accuracy: 0.14790996784565916
k= 8   accuracy: 0.14790996784565916
k= 9   accuracy: 0.14790996784565916
k= 10   accuracy: 0.14790996784565916
k= 11   accuracy: 0.14790996784565916
k= 12   accuracy: 0.14790996784565916
k= 13   accuracy: 0.14790996784565916
```

在 K 取前 78 时准确度都是 0.14，当 K 达到 79 时，准确度突变为 0.36 并保持。

```
k= 79   accuracy: 0.36012861736334406
k= 80   accuracy: 0.36012861736334406
k= 81   accuracy: 0.36012861736334406
k= 82   accuracy: 0.36012861736334406
k= 83   accuracy: 0.36012861736334406
k= 84   accuracy: 0.36012861736334406
k= 85   accuracy: 0.36012861736334406
k= 86   accuracy: 0.36012861736334406
k= 87   accuracy: 0.36012861736334406
k= 88   accuracy: 0.36012861736334406
k= 89   accuracy: 0.36012861736334406
k= 90   accuracy: 0.36012861736334406
k= 91   accuracy: 0.36012861736334406
k= 92   accuracy: 0.36012861736334406
k= 93   accuracy: 0.36012861736334406
k= 94   accuracy: 0.36012861736334406
k= 95   accuracy: 0.36012861736334406
k= 96   accuracy: 0.36012861736334406
k= 97   accuracy: 0.36012861736334406
```

然后，对于欧氏距离：

```
The Eulidean Distance:

k= 1    accuracy: 0.3665594855305466
k= 2    accuracy: 0.3665594855305466
k= 3    accuracy: 0.39228295819935693
k= 4    accuracy: 0.40836012861736337
k= 5    accuracy: 0.36977491961414793
k= 6    accuracy: 0.36977491961414793
k= 7    accuracy: 0.3633440514469453
k= 8    accuracy: 0.3440514469453376
k= 9    accuracy: 0.3633440514469453
k= 10    accuracy: 0.3858520900321543
k= 11    accuracy: 0.40192926045016075
k= 12    accuracy: 0.40192926045016075
k= 13    accuracy: 0.4212218649517685
k= 14    accuracy: 0.40514469453376206
k= 15    accuracy: 0.3954983922829582
k= 16    accuracy: 0.40836012861736337
k= 17    accuracy: 0.40192926045016075
```

图中是最大值的部分截图。

故取 K=13，对测试集进行分类的结果如下：

	senator carl krueger thinks ipods can kill you	fear
2	who is prince frederic von anhalt	joy
3	prestige has magic touch	joy
4	study female seals picky about mates	joy
5	no e book for harry potter vii	joy
6	blair apologises over friendly fire inquest	fear
7	vegetables may boost brain power in older adults	joy
8	afghan forces retake town that was overrun by taliban	fear
9	skip the showers male sweat turns women on study says	joy
10	made in china irks some burberry shoppers	joy
11	britain to restrict immigrants from new eu members	joy
12	canadian breakthrough offers hope on autism	joy
13	russia to strengthen its military muscle	joy
14	alzheimer s drugs offer no help study finds	joy
15	uk police slammed over terror raid	fear
16	no rejects police state claim	fear
17	oprah announces new book club pick	joy
18	smith can t be buried until hearing	joy
19	african nation hopes whoopi can help	joy
20	tourism lags in bush s hometown	joy
21	air france klm profit rises	fear

2、回归的优化过程展示

虽然对于相关系数有了一个评测表作为指标，但是手动复制粘贴难免麻烦，所以为了更方便，我还是在代码中加上了测试相关系数的部分。for 循环自动在范围内跑参数，得到的结果部分截图如下：

首先，对于曼哈顿距离：

```
The Manhattan Distance:

K= 1   Correlation: 0.0190031723846
K= 2   Correlation: 0.0143868789926
K= 3   Correlation: -0.0334744734016
K= 4   Correlation: -0.038438686798
K= 5   Correlation: -0.0340669034725
K= 6   Correlation: -0.00852810136087
K= 7   Correlation: 0.0286461480207
K= 8   Correlation: 0.0312812115766
K= 9   Correlation: 0.0317464703021
K= 10   Correlation: 0.0573852883749
K= 11   Correlation: 0.0103227815215
K= 12   Correlation: -0.0280076280928
K= 13   Correlation: -0.0205521996956
K= 14   Correlation: 0.0375246822684
K= 15   Correlation: 0.0444441732718
K= 16   Correlation: 0.0322412100243
K= 17   Correlation: -0.00160321176798
K= 18   Correlation: 0.000203745470156
K= 19   Correlation: 0.0100347149504
K= 20   Correlation: 0.0222712837791
K= 21   Correlation: 0.0241559189564
```

然后，对于欧式距离：

```
The Eulidean Distance:

K= 1   Correlation: 0.194795569458
K= 2   Correlation: 0.217257372897
K= 3   Correlation: 0.284957651843
K= 4   Correlation: 0.300745848516
K= 5   Correlation: 0.275628204767
K= 6   Correlation: 0.254985289081
K= 7   Correlation: 0.260211219426
K= 8   Correlation: 0.253072587976
K= 9   Correlation: 0.250651785408
K= 10   Correlation: 0.255174093211
K= 11   Correlation: 0.266355087714
K= 12   Correlation: 0.262752660824
K= 13   Correlation: 0.26327740007
K= 14   Correlation: 0.273807909915
K= 15   Correlation: 0.268341007642
K= 16   Correlation: 0.269820269659
K= 17   Correlation: 0.266872981316
K= 18   Correlation: 0.261594525104
K= 19   Correlation: 0.255533045904
K= 20   Correlation: 0.254258810942
K= 21   Correlation: 0.257776803523
```

K 的初始取值 4 即为最佳，即在测试集上的输出结果与前述相同

实验指标展示（优化后）

1、分类的实验指标分析

从结果中我们可以看出，曼哈顿距离衡量的准确度整体小于欧氏距离，故弃之不用。在欧氏距离测量情况中，当 K=13 时，准确度达到峰值 0.42。所以选用 K=13 的欧氏距离测量来对测试集进行分类操作。

2、回归的实验指标分析

从结果中我们可以看出，使用曼哈顿距离测量概率，基本在 0 附近波动，很多情况下还是负相关，比起使用欧氏距离，曼哈顿距离的相关系数实在是太低了。

对于欧氏距离，跑完一遍后电脑给出的最佳取值是：

```
The proper distance:Euclidean Distance
The proper K: 4
The max correlation: 0.300745848516
```

一开始的取值就碰巧到了一定范围内相关系数最高的情况了，最高值刚达低度相关的我本质也和要加油的小辣鸡没有区别呢。

思考题

1、IDF 的第二个计算公式中分母多了个 1 是为什么？

如果出现分母为 0 的情况，则 IDF 会转为无限大。分母多了个 1 的作用是防止出现分母为 0 的情况。

2、IDF 数值有什么含义？TF-IDF 数值有什么含义？

IDF 数值不仅衡量了词语的重要性，而且更加看重独有性。因为一个词在不同文本中出现的频率越大，IDF 反而越小。所以更加偏向这个词的特殊性。

TF-IDF 数值整体看重特殊的词语。某一文件中的词语出现更多+不同文件中词语出现更少的情况，整体侧重于某一文本中比较特殊的词语。

3、相似度加权为什么取距离为倒数？

这是基于 KNN 近邻算法的本质的，即距离越近越值得取的办法，将距离作为分母，距离越小，则权重更大。

4、各情感概率之和为 1，如何处理？

在本实验中，我在按照公式算出各概率后，将各概率除以所有情感的概率的和，则实现归一化操作。