中山大学移动信息工程学院本科生实验报告

（2015 年秋季学期）

课程名称：Artificial Intelligence　　任课教师：饶洋辉

年级	13 级	专业（方向）	移动信息工程
学号	13354485	姓名	朱琳
电话	13726231932	Email	280273861@qq.com

一.实验方法

1.阐述你的思路，并用公式，伪代码写出来

(1)首先我要对文本中的字符串进行全部读取并保存在数组中。然后我用 map<string,int>来标记一个单词是否重复出现，这样就会得到不重复的单词组。

伪代码：if map[word]=0 then no_repeat_word=word and map[word]++;

(2)同样是用 map 来对每个文本的的单词进行计数。

伪代码：for every text　map[word]++;

当读到每个文本末尾（如训练文本 1-训练文本 2）的时候，就进行存储。同时将 map 清空。这里的存储使用的是二维数组 v[i][j],其中 i 表示第 i 行文本（除了标题），j 表示对应的第 j 个不重复的单词。

(3)对于每个测试文本，将他们与每个训练文本的欧氏距离计算出来。并且存储欧式距离最小的那个文本的编号。然后将测试文本的感动值预测为此训练文档的。同时，若遇到有两个最小欧式距离的文档，那么取感动值较大的那个。

伪代码：

```
for 每个测试文档
    for 每个训练文档
        计算两文档的欧氏距离
        与目前最小的欧氏距离进行比较，并进行存储
    endfor
endfor
```

：

(4)对于实验最后一部分就是自由拓展方面，我主要测试了 city block 的方法和以各自欧式距离的倒数进行归一化之后作为权重的预测方式，此外我还尝试使用欧氏距离最小的前 k（k 为 1-10）个文本的

倒数作为衡量的标准（即进行排序，然后选取前 k 个 ， 需要归一化），当 k=1 时与计算欧氏距离的一样，k=10 的时候就相当于取所有的文本进行加权（即实验文档所说的加权方式）。

2.截图你的关键代码（注明使用的是什么语言）

【说明】使用 c++语言

【方法一】

```
    while (fin >> W1[num1])
        num1++;
    W1[num1++] = "EOF"; //增加冗余以确定文本的末端
    fin.close();

    //查找没有重复的单词
    clearMap(map, W1, num1);
    for (int j = 2; j < num1; j++) {
        int check = checkString(W1[j]);
        if (check == 0 && map[W1[j]] == 0) {
            W2[num2++] = W1[j];
            map[W1[j]] = 1;
        }
    }
```

```
//生成关系向量
int numText = 0;//numText指的是文本的数量即10+10=20
for (int i = 2; i < num1; i++) {
    int checkNow = checkString(W1[i]), checkNext = checkString(W1[i + 1]);
    if (checkNow == 0) {
        map[W1[i]]++;
    } else {
        clearMap(map, W2, num2);
    }

    if (checkNext != 0) { //读到下一个文本或者文件结束
        for (int j = 0; j < num2; j++) {
            v[numText][j] = map[W2[j]]; //第numText个文本对应第j个词的向量
        }
        numText++;
    }
}
```

```
//计算欧式距离
float dis, min; //训练文本和测试文本的关系
int pre[21] = { 0 }; //预测文本与训练文本最相关的文本编号
cout<<endl<<"训练文本和测试文本的欧氏距离表格:"<<endl;
cout<<"训练文本:  1    2    3    4    5    6    7    8    9    10"<<endl;
for (int i = 10; i < 20; i++) { //测试文本
    min = 9999999;
    cout<<"测试文本:";
    for (int j = 0; j < 10; j++) { //各个训练文本
        dis = 0;
        for (int k = 0; k < num2; k++)
            dis += (v[i][k] - v[j][k]) * (v[i][k] - v[j][k]);
        dis=sqrt(dis);
        printf("%.2f  ",dis);
        //cout<<setw(3)<<dis<<" ";
        if (min > dis) {
            min = dis;
            pre[i] = j;
        } else if (min == dis && moved[j] > moved[pre[i]]) {
            pre[i] = j;//遇到欧式距离相同的时候，取感动值大的那个作为预测值
        }
    }
    cout<<endl;
}

    fstream fout("E:/Desktop/人工智能/人工智能实验/Lab_1/result.txt");
    cout<<endl<<"以上测试文本的预测感动值为:"<<endl;
    for (int i = 10; i < 20; i++) {
        cout << moved[pre[i]] << " ";
        fout << moved[pre[i]] << " ";
    }
    fout.close();
```

【方法二】——比较于方法一的改动部分

```
//将关系向量进行归一化
for (int i = 0; i < numText; i++) {
    int num=0;
    for (int j = 0; j < num2; j++)
        num+=v[i][j];
    for(int j=0;j<num2;j++)
        v[i][j]/=num;
}
```

【方法三】city block

这个的实现只需要在欧氏距离的基础上，将计算欧式距离的那行代码改成 city block 的代码即可。即

```
for (int i = 10; i < 20; i++) { //测试文本
    min = 9999999;
    cout<<"测试文本:";
    for (int j = 0; j < 10; j++) { //各个训练文本
        dis = 0;
        for (int k = 0; k < num2; k++)
            dis += abs(v[i][k] - v[j][k]);// * (v[i][k] - v[j][k]);//city Block
        //dis=sqrt(dis);
        printf("%.2f  ",dis);
        //cout<<setw(3)<<dis<<" ";
        if (min > dis) {
```

【方法四】欧氏距离加权方法/city block 加权方法——对于每个测试文本，将相对于 10 个训练文本的欧式距离的倒数作为权重，然后进行归一化，再将对应 10 个训练文本的感动值乘以权重之后进行加和，作为此测试文本的感动值。具体实现只需要对于每个训练文本将欧式距离保存后进行权重的计算并且归一化即可。对于 city block 的加权只需要将计算欧式距离的代码改为 city block 即可。代码如下：

```
    float preMoved[100]={0};
    for (int i = 10; i < 20; i++) { //每个测试文本
        cout<<"测试文本"<<i-10<<": ";
        float dis[100],weight[100];//训练文本和测试文本的对应欧氏距离，以及距离的倒数
        int NoTexts=0;
        float num=0;
        float max_weight1=0,max_weght2=0;
        for (int j = 0; j < 10; j++) { //各个训练文本
            dis[NoTexts] = 0;
            for (int k = 0; k < num2; k++){
                dis[NoTexts] += (v[i][k] - v[j][k])* (v[i][k] - v[j][k]);
            }
            dis[NoTexts]=sqrt(dis[NoTexts]);//计算得出欧氏距离
            weight[NoTexts]=1.0/dis[NoTexts];
            num+=abs(weight[NoTexts]);//将每个训练文本权重相加，为归一化做准备
            NoTexts++;
        }
        //对于每个欧式距离的权重进行归一化
        cout<<"weight: ";
        for(int j=0;j<10;j++){//将本身权重/总权重
            weight[j]/=num;
            cout<<weight[j]<<" ";
        }

            //对于每个测试文本，将加权之后的感动值算出
            for(int c=0;c<NoTexts;c++){
                cout<<preMoved[i]<<" ";
                preMoved[i]+=weight[c]*moved[c];
            }
            cout<<endl;
            cout<<"move: "<<preMoved[i]<<endl;
    }
```

【方法五】kNN——即选取欧式距离比较大的前 k 位作为权重主要部分。后 10-k 位权重为 0。实现代码如下，出乎意料，这个的效果并不算好。在 k=4,5 附近相关性比较大，其他的时候都比较小。然而全部没有超过 k=10 的时候的相关性。可能因为数据集太小导致误差比较大吧。

```
float preMoved[100]={0};
for (int i = 10; i < 20; i++) { //每个测试文本
    cout<<"测试文本"<<i-10<<": ";
    float dis[100];//训练文本和测试文本的对应欧氏距离，以及距离的倒数
    weight_and_No weight[100];
    int NoTexts=0;
    //float num=0;
//  float max_weight1=0,max_weght2=0;
    for (int j = 0; j < 10; j++) { //各个训练文本
        dis[NoTexts] = 0;
        for (int k = 0; k < num2; k++){
            dis[NoTexts] += (v[i][k] - v[j][k])* (v[i][k] - v[j][k]);
        }
        dis[NoTexts]=sqrt(dis[NoTexts]);//计算得出欧氏距离
        weight[NoTexts].weight=1.0/dis[NoTexts];
        weight[NoTexts].No=NoTexts;
        NoTexts++;
    }
    sort(weight,weight+NoTexts,cmp);
    float num=0;
    int cut=2;//取权重最大的前cut个进行归一化加权
    for(int z=0;z<cut;z++){//取前三个最大权重的作为总体权重
        num+=weight[z].weight;
    }
    for(int z=0;z<cut;z++){
        weight[z].weight/=num;
    }
```

二. 实验结果

1.列出每种方法运行的结果

(1)方法 1

```
没有重复的单词数=49

消防员 冲进 火场 救出 男童 公务员 患 癌症 保持 在岗 多次 救人 不幸 身亡 老人 成
功 进行 免费 白内障 手术 海豚 误 吞 排球 后 取出 6旬 跳楼 自杀 男子 疑犯 枪杀 出
租车 司机 妻子 医师 把 肾脏 当 肝脏 致人 喝药 清醒 将 搭载 杀 弟媳 患者 捅破
```

```
训练文本1=<1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本2=<0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本3=<1 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本4=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本5=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本6=<0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本7=<0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本8=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本9=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本10=<0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 >
测试文本1=<0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
测试文本2=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 1 0
 0 0 0 0 0 0 1 0 0 0 0 0 0 0 >
测试文本3=<0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 1 0 0 0 0 0 0 >
测试文本4=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
测试文本5=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
测试文本6=<1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 >
测试文本7=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 >
测试文本8=<0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 1 1 0 0 >
测试文本9=<0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 1 0 0 1 1 0 0 0 0 0 0 1 1 >
测试文本10=<0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
```

```
训练文本和测试文本的欧氏距离表格:
训练文本:  1     2     3     4     5     6     7     8     9     10
测试文本:3.00  2.65  3.32  2.00  2.83  2.65  2.83  2.83  3.00  3.46
测试文本:3.32  3.32  3.61  3.46  3.46  3.00  2.45  2.45  1.73  3.74
测试文本:3.16  2.45  3.46  3.00  3.32  3.16  3.00  3.00  3.16  3.32
测试文本:2.83  2.83  3.16  3.00  3.32  2.00  1.00  2.65  2.00  3.32
测试文本:3.16  3.16  3.46  3.00  3.32  2.45  2.24  2.65  1.41  3.61
测试文本:1.00  3.32  2.65  3.46  3.74  3.32  3.16  3.16  3.32  3.74
测试文本:3.16  3.16  3.46  2.65  3.61  2.83  3.00  2.24  3.16  3.61
测试文本:3.46  3.46  3.46  3.61  3.32  2.83  2.24  3.32  2.45  3.32
测试文本:3.61  3.61  3.61  3.74  3.74  3.32  3.16  3.46  3.61  2.00
测试文本:3.00  3.32  2.24  3.16  3.74  2.24  2.83  3.16  3.32  3.46

以上测试文本的预测感动值为:
0.5 0 0.9 0.02 0 1 0 0.02 0 0.5
```

```
请依次输入10篇测试文本的预测结果，以空格或回车分隔:
0.5 0 0.9 0.02 0 1 0 0.02 0 0.5
你的上述预测结果与标准答案的相关系数(-1到1之间)为:
0.8745725106173748
```

（2）归一化之后

```
训练文本1=<0.2 0.2 0.2 0.2 0.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本2=<0 0 0 0 0 0.2 0.2 0.2 0.2 0.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本3=<0.142857 0.142857 0.142857 0 0 0 0 0 0 0 0.142857 0.142857 0.142857 0
.142857 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本4=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.166667 0.166667 0.166667 0.166667 0.166
667 0.166667 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本5=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.125 0 0 0 0.125 0.125 0.125 0.125 0.1
25 0.125 0.125 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本6=<0 0 0 0 0 0 0 0 0 0 0 0 0 0.2 0.2 0 0 0 0 0 0 0 0 0 0 0 0.2 0.2 0.2 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本7=<0 0 0 0 0 0 0 0 0 0 0 0 0 0.25 0 0 0 0 0 0 0 0 0 0 0 0 0 0.25 0.25 0.
25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本8=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.25 0.25
 0.25 0.25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本9=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.2 0 0 0 0.2 0.2 0 0
.2 0 0 0.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
训练文本10=<0 0 0 0 0 0 0 0 0 0 0 0 0 0.125 0 0 0 0 0 0 0 0.125 0 0 0 0 0 0 0 0
0 0 0 0 0 0.125 0.125 0.125 0.125 0.125 0.125 0 0 0 0 0 0 0 0 >
测试文本1=<0 0 0 0 0 0 0 0.25 0 0 0 0 0 0 0.25 0.25 0 0 0 0.25 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
测试文本2=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.166667 0 0 0 0.1666
67 0.166667 0 0.166667 0 0.166667 0 0 0 0 0 0 0 0.166667 0 0 0 0 0 0 0 >
测试文本3=<0 0 0 0 0 0 0 0.2 0.2 0 0 0 0 0 0 0 0 0 0 0.2 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0.2 0 0 0 0 0 0 0.2 0 0 0 0 0 0 >
测试文本4=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.333333 0.3333
33 0.333333 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
测试文本5=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.2 0 0 0 0 0 0 0 0 0 0.2 0 0 0 0.2 0.2 0
 0.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
测试文本6=<0.166667 0.166667 0.166667 0.166667 0.166667 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.166667 0 0 0 0 0 >
测试文本7=<0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.2 0 0 0.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
.2 0.2 0 0 0 0 0 0 0 0 0 0 0.2 0 0 0 0 >
测试文本8=<0 0 0 0 0 0 0 0 0 0 0 0 0 0.142857 0 0 0 0 0 0 0 0.142857 0 0 0.14285
7 0 0 0 0.142857 0.142857 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.142857 0.142857 0 0 >
测试文本9=<0 0 0 0 0 0 0 0 0 0 0 0 0 0.125 0 0 0 0 0 0 0 0.125 0 0 0 0 0 0 0 0 0
 0 0 0 0 0.125 0.125 0 0 0.125 0.125 0 0 0 0 0 0 0.125 0.125 >
测试文本10=<0 0 0.166667 0 0 0 0 0 0 0 0 0.166667 0.166667 0.166667 0.166667 0 0
 0 0 0 0 0 0 0 0 0 0.166667 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 >
```

```
训练文本和测试文本的欧氏距离表格:
训练文本:  1     2     3     4     5     6     7     8     9     10
测试文本:0.67  0.59  0.63  0.41  0.50  0.59  0.71  0.71  0.67  0.61
测试文本:0.61  0.61  0.56  0.58  0.50  0.55  0.50  0.50  0.32  0.54
测试文本:0.63  0.49  0.59  0.55  0.52  0.63  0.67  0.67  0.63  0.52
测试文本:0.73  0.73  0.69  0.71  0.68  0.52  0.29  0.76  0.52  0.68
测试文本:0.63  0.63  0.59  0.55  0.52  0.49  0.50  0.59  0.28  0.57
测试文本:0.18  0.61  0.41  0.58  0.54  0.61  0.65  0.65  0.61  0.54
测试文本:0.63  0.63  0.59  0.48  0.57  0.57  0.67  0.50  0.63  0.57
测试文本:0.59  0.59  0.49  0.56  0.44  0.48  0.42  0.63  0.41  0.44
测试文本:0.57  0.57  0.48  0.54  0.47  0.52  0.56  0.61  0.57  0.25
测试文本:0.55  0.61  0.35  0.53  0.54  0.41  0.58  0.65  0.61  0.50

以上测试文本的预测感动值为:
0.5 0 0.9 0.02 0 1 0.5 0 0 0.5
```

```
请依次输入10篇测试文本的预测结果，以空格或回车分隔:
0.5 0 0.9 0.02 0 1 0.5 0 0 0.5
你的上述预测结果与标准答案的相关系数<-1到1之间>为:
0.9108972268655023
```

（3）方法三——city block

```
以上测试文本的预测感动值为:
0.5 0 0.9 0.02 0 1 0 0.02 0 0.5
```

```
onCorr
请依次输入10篇测试文本的预测结果，以空格或回车分隔:
0.5 0 0.9 0.02 0 1 0 0.02 0 0.5
你的上述预测结果与标准答案的相关系数<-1到1之间>为:
0.8745725106173748
```

（4）方法 4.1 欧式距离转权重

```
以上测试文本的预测感动值为:
0.357578 0.301192 0.358532 0.284093 0.294428 0.466551 0.335572 0.312288 0.308772
 0.345016
```

```
请依次输入10篇测试文本的预测结果，以空格或回车分隔:
0.357578 0.301192 0.358532 0.284093 0.294428 0.466551 0.335572 0.312288 0.308772
 0.345016
你的上述预测结果与标准答案的相关系数<-1到1之间>为:
0.9207294778112801
```

（5）方法 4.2 city-block 转化为权重（这个是目前实现最好的）

```
以上测试文本的预测感动值为:
0.363662 0.262848 0.374552 0.247442 0.229631 0.558413 0.32365 0.289921 0.262737
 0.343423
```

```
onCorr
请依次输入10篇测试文本的预测结果，以空格或回车分隔:
0.363662 0.262848 0.374552 0.247442 0.229631 0.558413 0.32365 0.289921 0.262737
 0.343423
你的上述预测结果与标准答案的相关系数<-1到1之间>为:
0.9271688373168595
```

（6）kNN——欧式距离转化实现的效果不太好，无论 k 取 1-9 的何值都不如用总体的那个效果好。

City-block 中 k=4 的时候有个效果稍微不错的，贴出来

2.实验结果的简要比较分析

【分析 1】我们看到归一化后的相关性比归一化之前的相关性好些，对比上述方式的感动值的预测：

归一化之前：

归一化之后：

我们看到对于第 7 个和第 8 个的测试文本的感动值的估测两种方式存在差异，即

	测试文档 7，出租车 司机 免费 搭载 老人	**测试文档 8，男子 误 杀 弟媳 后 自杀 身亡**
归一化之前	0，训练文档 8，疑犯 枪杀 出租车 司机	0.02，训练文档 7，男子 跳楼 自杀 身亡
归一化之后	0.5,训练文档 4，老人 成功 进行 免费 白内障 手术	0，训练文档 9，男子 枪杀 妻子 后 自杀

归一化的好处的消除数据本身的大小对实验结果造成的影响，我发现，这些词组的长度对于他们的预判其实起到比较重要的作用，如果对这些向量进行归一化加权之后，那么就可以消除词组的多少对于欧氏距离计算的影响，判断的准确率自然就会提高。

【分析 2】这周实现了实验的后半部分。后面我所用到的几种方法中是用 city block 距离的倒数，归一化之后作为权重的效果是最好的，但是跟用欧式距离的相差不大，但是我感觉应该是样本的问题，毕竟现在数据挺少的，单单靠这些样本现在根不能准确的说出那个比较好，但根据我自己的一些理解和思考，我感觉以后用欧氏距离的倒数作为权重比较好，毕竟 city block 好像受到文本字数的牵连挺大的。

【分析 3】衡量全部 VS kNN

前提是最后的权重都要归一化。其实在做实验之前我感觉是后者实现的效果会比较好，但是实验结果却让我有点失望，如果有很大的数据的话后者一定比前者好。由于本次实验总体数量实在太小，所以选取的样本的数量对实验结果有很大的影响,比如无论选 city block 还是欧氏距离的倒数作为权重，当 k=4,5 的时候的效果是好于其他的 k 值的。但最终都没有超过使用总体数据的相关性。