# 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首先我需要读取文本文件和每个训练文本的情感值。由于每个文本都有一些固定的属性，所以我需要一个数据结构将他们聚合起来，以文本为单位。
②得到不同的单词（这个在读文本的时候可以直接判断）以及计算每个文本的向量，并将他们归一化。

③计算每个测试文本与每个训练文本之间的距离（可以是欧式距离，也可以是 city-block 距离或者其他。

④根据 kNN 算法，需要手动输入 k。然后我们对于每个测试文本需要找到与之距离最小的 k 个训练文本。

⑤ 将这 k 个文本距离进行取倒数，然后将这些权重归一化。最后将这些权重乘以他们的情感值即为测试文本情感值。

### 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；

①关于文本的数据结构。k_th 为距离最近的 k 个训练文本的一些参数。

```
struct Files { //每个训练文本
    string words[100]; //文本的所包含的单词
    float vector[5000]; //与不相同的单词对应生成的向量
    float feel[10];
    float dis[2000]; //每个测试文本与各个训练文本的距离。训练文本不需要这个。
    float num;
};
struct k_th {
    float dis;
    float probability;
    float weight;
};
```

②计算向量。以及向量的归一化

```
void get_vector() {
    for (int row = 0; row < num_of_trains; row++) { //对于每个训练文本
        trains[row].num = 0;
        for (int col = 0; col < num_of_unique; col++) {
            if (map_train[row][(unique_words[col])] == 1) {
                trains[row].vector[col] = 1;
                trains[row].num++;
            } else {
                trains[row].vector[col] = 0;
            }

        }
    }

    for (int row = 0; row < num_of_tests; row++) { //对于每个测试文本
        tests[row].num = 0;
        for (int col = 0; col < num_of_unique; col++) {
            if (map_test[row][(unique_words[col])] == 1) {
                tests[row].vector[col] = 1;
                tests[row].num++;
            } else
                tests[row].vector[col] = 0;
        }
    }
}
void vector_to_one() {
    for (int row = 0; row < num_of_trains; row++) {
        for (int col = 0; col < num_of_unique; col++) {
            trains[row].vector[col] /= trains[row].num;
        }
    }

    for (int row = 0; row < num_of_tests; row++) {
        for (int col = 0; col < num_of_unique; col++) {
            tests[row].vector[col] /= tests[row].num;
        }
    }
}
```

③ 得到距离。最终我选用的是 city'-block 距离，这个跑出来的效果最理想。

```
void get_dis() {
    for (int t_row = 0; t_row < num_of_tests; t_row++) { //每个测试文本
        for (int train_row = 0; train_row < num_of_trains; train_row++) { //每个训练文本
            tests[t_row].dis[train_row] = 0;
            float distance = 0;
            for (int k = 0; k < num_of_unique; k++) {
//              distance +=
//                      (trains[train_row].vector[k] - tests[t_row].vector[k])
//                              * (trains[train_row].vector[k]
//                                      - tests[t_row].vector[k]);
                distance +=
                        abs(trains[train_row].vector[k] - tests[t_row].vector[k]);
            }
            //tests[t_row].dis[train_row] = sqrt(distance);
            tests[t_row].dis[train_row] =distance;
        }
    }
}
```

④kNN

```
void caculate(int k, k_th v[], int f, Files &test) { //取倒数，算权重，权重归一化，计算anger等值
//  float num_dis = 0;
    test.feel[f] = 0;
//  for (int i = 0; i < k; i++) {
//      num_dis += v[i].dis;
//  }
    float num = 0;
    for (int i = 0; i < k; i++) {
        //v[i].dis /= num_dis; //先进行向量的归一化
        v[i].weight = 1.0 / v[i].dis;
        num += v[i].weight;
    }
    for (int i = 0; i < k; i++) {
        v[i].weight /= num;
        test.feel[f] += v[i].weight * v[i].probability;
    }
}
```

```
void kNN(int k, int f) {
    fstream out;
    if (f == 0)
        out.open(name1);
    else if (f == 1)
        out.open(name2);
    else if (f == 2)
        out.open(name3);
    else if (f == 3)
        out.open(name4);
    else if (f == 4)
        out.open(name5);
    else if (f == 5)
        out.open(name6);
    out.clear();
    for (int t_row = 0; t_row < num_of_tests; t_row++) { //对于每个测试文本
        //得到距离此test最近的k个向量。
        k_th v[300];        //距离此test最近的文本。
        int number[300];
        for (int count = 0; count < k; count++) {       //计算距离测试文本最近的k个文本;
            float min = INT_MAX;
            for (int train_row = 0; train_row < num_of_trains; train_row++) {
                if (tests[t_row].dis[train_row] < min) {
                    min = tests[t_row].dis[train_row];
                    number[count] = train_row;      //第trains_row个训练文本
                }
            }       //循环找到最小值。
            v[count].dis = min;
            v[count].probability = trains[(number[count])].feel[f];
            tests[t_row].dis[(number[count])] = INT_MAX;//将此轮找到的那个最小距离置为最大，不再参与比较。共有k轮比较
        }
        for (int count = 0; count < k; count++) {
            tests[t_row].dis[(number[count])] = v[count].dis;       //数据恢复
        }
```

```
        //至此已得到距离此test最近的k个向量。
        caculate(k, v, f, tests[t_row]);
        cout << tests[t_row].feel[f] << " ";        //输出预测值。
        out << tests[t_row].feel[f] << endl;        //输出预测值。
    }
    out.close();
}
```

## 二. 实验结果

①anger

```
E:\Desktop\lab2\AILab>java -Xmx256M -classpath ./AILab.jar Lab2
请输入anger, disgust, fear, joy, sad, surprise中的任何一种进行评测:
anger
请确保在predict_test文件夹里已经有anger_predict.txt文件!
你的上述预测结果与标准答案的相关系数(-1到1之间)为:
0.21195186813677064
```

②disgust

```
E:\Desktop\lab2\AILab>java -Xmx256M -classpath ./AILab.jar Lab2
请输入anger, disgust, fear, joy, sad, surprise中的任何一种进行评测:
disgust
请确保在predict_test文件夹里已经有disgust_predict.txt文件!
你的上述预测结果与标准答案的相关系数(-1到1之间)为:
0.1832598456833196
```

③fear

```
E:\Desktop\lab2\AILab>java -Xmx256M -classpath ./AILab.jar Lab2
请输入anger, disgust, fear, joy, sad, surprise中的任何一种进行评测:
fear
请确保在predict_test文件夹里已经有fear_predict.txt文件!
你的上述预测结果与标准答案的相关系数(-1到1之间)为:
0.301425080930117
```

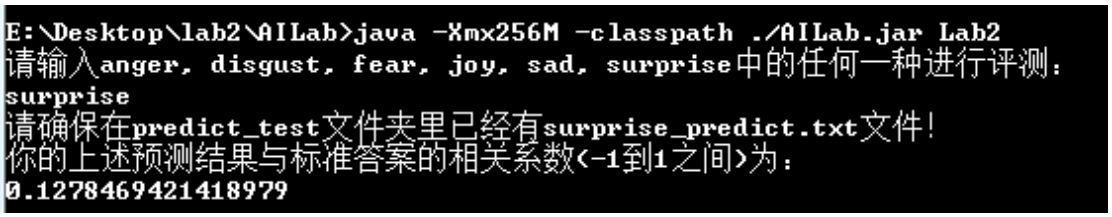
④joy

```
E:\Desktop\lab2\AILab>java -Xmx256M -classpath ./AILab.jar Lab2
请输入anger, disgust, fear, joy, sad, surprise中的任何一种进行评测:
joy
请确保在predict_test文件夹里已经有joy_predict.txt文件!
你的上述预测结果与标准答案的相关系数(-1到1之间)为:
0.22490708954800342
```

⑤sad

```
E:\Desktop\lab2\AILab>java -Xmx256M -classpath ./AILab.jar Lab2
请输入anger, disgust, fear, joy, sad, surprise中的任何一种进行评测:
sad
请确保在predict_test文件夹里已经有sad_predict.txt文件!
你的上述预测结果与标准答案的相关系数(-1到1之间)为:
0.18024871482289592
```

⑥surprise




**【结果分析】**最终的实现还是不算太好。时间有点紧张，平均值大概是 0.205。不过现在我还没有添加 stopwords，我打算让程序通过学习文本，自己把 stopwords 生成出来，代码可能要多一点。下次实验之前肯定可以实现了。