INDOOR LOCALIZATION

EEC 687:-MOBILE COMPUTING
FINAL REPORT

DHARIN PATEL
&
VARIL PATEL

PROFESSOR
DR. CHANSU YU
Abstract

Use of mobile computing and wireless LAN increased day by day so using indoor localization we can track the location of user inside closed building. Here, we present fingerprint base indoor localization technique, for this we have used training dataset and test dataset as a fingerprint and compare signals strength of appropriate access point with each other. We also present our calculation and technique to find accurate position of object to minimize estimation error.

1. Introduction

Increased use of mobile devices and wireless LAN encourage for locating particular object in space. Use of this locating system varies from person to person, but somehow in big mall or in college if we want to find specific store or class then we can make our task easy using indoor localization system because GPS is not applicable in this case due to close environment. Even if we want to find one person in building so using this technique we can reach to his or her location.[1]

The components for the system are communication network, mobile device and positioning device, in this system the basic function is transmitter send the signal and receiver receive the signal, here the mobile phone is receiver, the signal is coming via communication network and mobile phone receive the signal and we can find the location of the device.

In figure 1, we can show you clearly indoor localization system. A person who has mobile device found in the office, the mobile device receives the signal via router and it gives the location of the device. This is simple procedure of indoor localization system. Indoor localization provides a positioning for closed structure like hospitals, airport, university campus, shopping mall etc....there are some challenges like
walls, movement of humans, door, furniture, equipment and other factors. Multiple signals are reflected by these factors so we cannot identify the exact position because the electromagnetic waves propagate by these additional factors.

In this paper, we present finger print base indoor localization system and to find accurate location of person, we have explained nearest neighbor method and use of weighted distance method, here we have used nearest neighbor method for all calculation between training dataset and test dataset. We use signal strength which are collected at different location, we have location of object and according co-ordinates for each fingerprints and used those for calculating Euclidian distance between object’s location in training and test dataset and using nearest neighbor method we calculated accurate position of object and tried to reduce estimation error.

Following paper is distributed in different section. In section 2 we discuss related work for location methods, in section 3 we mention research methodology section 4 contain algorithm and calculation to improve accuracy for locating object and finally, we conclude our paper.

2. Related work for location methods

There are many localization techniques like GPS, laser scanning and radio frequency identification and detection, but they are not good enough for indoor localization. Laser scanners are used on particular object to find the location information objects, laser scanning takes high resource utilization like time, CPU and memory. RFID tags are attached to object to collect radio signal data to find the object’s location.

There are technical approaches applied to indoor localization: indoor global positioning, computer vision, time-of-arrival and received signal-strength index.

In active badge system, badge is mounted on particular object and emit infrared signal signals while sensor receive this signal and gives location, it gives location of object but it faces problems related to short range of infrared signal, installation of particular hardware and it performance is poor in presence of sunlight, instead of using infrared if we use RF network then we can get more range and scalability. In RADAR paper they show one experiment in which base stations are placed on different location of indoor building which are in open area without any obstacle, in mobile devices like computer we have network interface card which gives information about signal strength, to find exact and nearest matches we need particular algorithm like nearest neighbor method and when we get object near to access point that time we get strong signal strength [1].

Signal transmission base stations are fixed in fix locations, it produces signals at a fix frequency and the receiver gets signals from transmitters, it can determine its location through several techniques as follow:

1) Triangulation

It is used geometric properties of triangles to find location of object. If mobile device receives the Wi-Fi signals from many Wi-Fi access points so that time received signal strength, angle of arrival and the time of arrival used to find the distance of object from Wi-Fi access point using this method.
2) Proximity

When mobile device receive Wi-Fi signal from various Wi-Fi access points at target location, at that time one Wi-Fi access point gives strongest signal strength so location of that access point is considered as location of object. In short this method use proximity between Wi-Fi access points and target place [3].

3) Fingerprint Based Indoor Localization

The main function of Fingerprinting Based Indoor Localization is to collect finger print from the surrounding location and matched with database. There are two phase in the Fingerprinting Based Indoor Localization. One is phase 1 which is offline and second is the localization phase which is online.[2]

Finger print based location detection is different than particular model based location because model based approach use distance between transmitter and receiver to find the location of particular object also use geometrical calculation like time of arrival and the difference of arrival ,but for indoor building it is difficult to create some model due to obstacles and it gives less accuracy, while signal travel it get some types of reflections and multiple paths so it effect on finding accurate location.

Dividing all finger prints in cluster to reduce complexity and RSS gives same behavior in each cluster, principal component analysis (PCA) is used to mitigate noise and dimensions for fingerprints it use orthogonal transformation and decrease number of variables. In positioning system PCA is used to transform data from offline and online fingerprints [6].

PCA makes set of unrelated variables which are called principle components. PC can be calculated by subtracting mean RSS values of all reference point and mean RSS value of particular reference point as follow

\[ Xi = Ai - A' \]

Ai= mean RSS value for particular point

A' = mean RSS value for all reference point

Principle component matrix can be express by \[ Zi = Xi*E \] Where E= Eigen vector matrix for particular reference point

After arranging all principle components decreasing order, we can reduce matrix dimension by selecting first n principal components from all availability.

Where n<=dimension of input RSS values
**Indoor Localization**

**Affinity propagation clustering**

In indoor localization we got reflection and fading effect of Wi-Fi signal because of wall and other items of indoor building to reduce all time effect AP algorithm use.

Using this cluster is made and which gives similarity of RSS value in cluster and using ANN model we can apply on each cluster to find accurate poison.

Here similarity between signal strength can be represented by following as

\[ S = |A - B| \]

where A and B is different fingerprints,

To make proper cluster following is used

\[ S[i,j] = \min\{s(i,j)\} \]

Using above method, fixed point is made from each cluster and during online phase compare figure point with fixed point to choose particular cluster. In that way it increase accuracy and decrease time delay because it just compare with fingerprint which belongs to cluster.

After making clusters using AP algorithm model can be made to replace traditional fingerprint map and find relationship between received signal strength and position.

Clustering is not accurate in large set of access point, if we miss one or more APs during comparison then it results in wrong Euclidean distance and gives wrong sets of fingerprint.

Fingerprint based positioning system takes long time because it compares fingerprint from test data set with each and every training data set. Using Affinity propagation it makes cluster with RSS values which share same features and ANN provide higher localization accuracy [6].

**Representative Access points**

In normal way of Wi-Fi based localization algorithm compare each fingerprint from test data set with all available fingerprint in training data set so it will affect on accuracy as well as it will take long time for comparison as well create more traffic.

If we use passive scanning and unicast request and also destination access point are known then we can avoid all above problems. if client device know the roughly the area of device then using representative access points request directed to particular part of fingerprints in dataset instead of comparing with each fingerprints.

In this representative access point method entire indoor area is divided in segments using intersections and few representative access point are mentioned for particular segment using following criteria 1) APS are observed in whole segment 2) it has high received signal strength value to avoid fluctuation 3) each APS are different from each other. Using APS we can find best match area FPS in terms of signal strength .RSS of some APS observed during offline stage to find wall attenuation factor and loss export in path. Using that we can find the position of client device in online phase [7].

Path loss distance=

\[ d_0 + 10n \log (d/d_0) + p*X + q*Y \]
do= reference distance  
n=path loss exponent  
p=number of soft walls  
q=number of concrete walls  
X= attenuation factor of soft wall  
Y= attenuation factor of concrete wall  

3. Research Methodology  
We have training dataset and test dataset which are measured at Hong Kong University of Science and Technology indoor building whose area is 145.5m ×37.5m and whole area is divided in different grids and each grid has specific size of 1.5m ×1.5m so in this way whole area is divided in 247 grids so in whole map each grids are connected with each other and all have numbers.[5]  

![Area of building where training and test data collected](image)

We have different technique to find the indoor localization position. The best one is which gives more accurate result so researcher already used in past and they got some result. In 2003, using the radar system the measurement of location error found 2.1m and after that it is used lease system for the finding location which gives system accuracy 2.1 meters for same area. One more method MD which gives accuracy 1.3m. After some year they used other technique for new location which is Euclidean distance and it was give 4.47m and if they used fuzzy logic than they got 3m. In 2008, KNN algorithm method we got 2m. And same things but additional predict KNN algorithm and they got 1.3m. Using PKNN algorithm finally they got 33% of improve accurate result. finally 2013, fingerprint positioning system calculate the error between target places to mobile device which gives 2m which is most accurate result so we used finger print based technique for our project[8].  

Finger print based indoor localization system contain two steps to complete whole process. These two steps are: off line phase and online phase. Here, we have training and test dataset so we can consider training dataset as a part of offline phase and test dataset as a part of online phase.  

**Offline phase:** it is used to make radio map by observing signal strength for current location of object from different access points. In this way, at particular location object receive signal strength from different
access points and its value varies between 0 to -99dBs so for different location we get different signal strength from different access points all these different locations give different fingerprints and all are stored in data base for considering it as a reference for matching with real time fingerprints. In our case we have training data for different location with reference to different access points and each fingerprint has different location and signal strength values. Here we have 2322 lines of data which means we have that much fingerprints to consider as a radio maps and total access points are 99 and 247 different location.

In following figure follow this data structure:

```
<Location Label><AP's Index_1><RSS value><AP's Index_2><RSS value><AP's Physical Index><RSS value>
```

Here in some fingerprint location label is -1 it means that fingerprint has unknown location.

![Figure:4 fingerprints of training data sets](image)

**Online Phase:** it is used to find location of device, here we get particular location of object and based on that location we get signal strength from different location all these give one fingerprint so we can compare this fingerprint with all present fingerprints in data base in our case it is training dataset, so here we have test data and each fingerprint has location so we take one fingerprint from test dataset and compared it with all available fingerprints in training dataset but here we skip those fingerprints whose location is -1 means unknown, using this we can find estimation location is the fingerprint which has the minimum distance among fingerprints in the radio map. Here in test data set we have 53 fingerprints and we take difference of RSS values between test data set and training data set for available access points and then we found minimum distance for that pair, if particular access point is not available in training data set compare to test then we took -99dBs signal strength for that access point.
In this way if we get fingerprint from current location of user f' and compare with database fingerprint f then we get RSS difference d=|f-f'| using this we can find ||d||. If we match all these fingerprint in this way then we can find the fingerprint f’ that achieve the highest match with the query fingerprint f. After following all above steps the user’s location is estimated as the corresponding location L(f*) of f* and by assuming we can get true location of f is L(f),so location estimation error is given by e=||L(f)-L(f*)||.[4]

4. Algorithm and implementation

We mention several algorithm for tracking user. In our case we have training dataset and test, to locate particular object first we take particular finger print from test dataset and compare it with all training dataset finger print based on signal strength value.

Idea of finding distance can be divided as follow

1) Signal strength sample is collected from fix access point.
2) Here localization of object is finding by matching signal strength with other and get best match, empirical method use signal strength information and location information area.
3) We have to use methodology to compare all fingerprint with measured one to get best matches signal strength
Our aim is to find distance between the observed sets location and location of recorded one, in our case we take test data set as observed and training as a recorded and we find specific location which gives minimum distance between them. In this calculation we used Euclidian distance method: $\sqrt{(ss1-ss1')^2+(ss2-ss2')^2+(ss3-ss3')^2+...}$.\[1\]

Implementation of Online and Offline phase in JAVA:-

As discussed above we have training and test data base and each has different fingerprints so manually process become more time consuming and it may give wrong values so we have developed java program (in Appendix) which compare each fingerprint of training data with test data and as discussed above we have used Euclidian distance method for both fingerprints.

We have assigned different buffer for storing training data set and test data set so each time first fingerprint of test data is taken and compared with all available fingerprints in training buffer. Here in addition to make process simple we have avoided those line of fingerprints from training data set which has -1 location because it indicate absence of location so it is useless for our calculation.

Fingerprint of training data set is compared with fingerprint of test data set and take difference according to appropriate access point and take square of all difference particularly and add all those and take square root which gives appropriate value for that RSS comparison if all available access points match with each other and if match not found for access point then array takes -99dBs value referring to that access point.

We compare training data and test data but it gives plenty of results in our case we need least value so we have added additional part to arrange all result in ascending order and generate one separate output also here we compared all 53 lines of test data with training data and we took 53 set of pair of training and test data set which gives minimum value for each test data set line and we have stored it in another file.

Output of Program and calculation:

As discussed in developed java program, comparison of each fingerprint of test data is done with all available fingerprint of training data set and just avoid those fingerprints whose locations are unknown. We get output as per following for all 53 dataset of test data. In the Figure 6, we got output and which show that minimum error between test data and training data set also it gives in ascending order of value so we got best closest fingerprint among all data sets. In that way we get output for all fingerprints and in figure 7 we get minimum error between each of fingerprint of test data set and training data set.
We have took particular test line to do calculation and it is


After running java code it gives appropriate RSS difference with training lines and least value is refers to following training line
INDOOR LOCALIZATION

Using di=|fi-fi’|

RSS difference={((-93+96)^2+(-87+83)^2+(-91+91)^2+(-86+84)^2+(-93+93)^2})^{1/2}=5.385

As per given co-ordinates for particular location 183 has (102, 36) and 220 has (97.5, 36)

So using e=||L(f)-L(f*)|| and triangulation method distance between these points can be find

{(102 − 97.5)^2 + (36 − 36)^2}^{1/2} = 4.50m It is called estimation error

As shown in output we took first five fingerprints of training dataset which closely match with RSS values of test dataset.

F1=220(97.5, 36) 18:-96 23:-83 45:-87 48:-87 62:-91 66:-84 67:-93 its d1=4.5
F2=125(103.5, 36) 18:-96 23:-83 45:-87 48:-87 62:-91 66:-84 67:-93 its d2=1.5
F3=135(106.5, 36) 18:-96 23:-83 45:-95 48:-95 62:-93 66:-88 67:-97 its d3=4.5
F4=232(108, 36) 18:-96 23:-83 45:-95 48:-95 62:-93 66:-90 67:-93 its d4=6
F5=178(105, 36) 18:-96 23:-83 45:-94 48:-93 62:-96 66:-86 67:-93 its d5=3

Here d1 to d5 is distance between location of test dataset’s fingerprint and training dataset fingerprint.

As per above we found closest fingerprints so here estimation error is as follow: 4.5, 1.5, 4.5, 6,3 using 

e=||L(f)-L(f*)|| and triangulation method.
Our aim is to reduce minimum estimation error to increase accuracy so here we have tried with nearest neighbor method.

Nearest neighbor method

In basic calculation we consider only one nearest neighbor in space to find distance ,but if we consider K nearest neighbors then many neighbors are roughly same distance from target point so signal strength is also closely match in that case we cannot avoid all other points by choosing one point. Here every neighbor has different space vector compare to neighbor if we do averaging then it gives new location closer to actual locations if value of K increase then accuracy decrease because averaging gives new location point which is far away from true location and they disturb estimation, in nearest neighbor method we consider those fingerprints whose RSS closely match with each test dataset. There are many fingerprints available, here we take first five closest fingerprints. We used nearest neighbor method in our implementation.

There are many points available which are at same distance from target location and here in each fingerprint we get variation in signal strength value so we cannot pick only single point from all because all are close to each other, location of every point is different so if we take average of co-ordinate then we can bring point more close to the true location.

If we take average of all first 5 locations (coordinates) of fingerprints then we will get [(97.5+103.5+106.5+108+106)/5, (36+36+36+36+36)/5]= (104.1, 36) now if we take this location as a guessed new location and location of 183 is true location so using

{(104.1 − 102)^2 + (36 − 36)^2}^{1/2}=2.1 m
INDOOR LOCALIZATION

Which gives 2.1m estimation error so it improves accuracy compare to above discussed method in same way if we take first three then it gives

\[
\frac{(97.5+103.5+106.5)}{3}, \frac{(36+36+36)}{3} = (102.5, 36),
\]

if we find distance between this location and 183 location then it gives

\[
\sqrt{(102.5 - 102)^2 + (36 - 36)^2} = 0.5m
\]

0.5 m estimation error if we take first 2 and 4 according fingerprints and used average then we get estimation error bigger than 0.5 so to improve accuracy if we take average of most closest three fingerprints location and get new location then this new location will help in improving accuracy. Using same process we find results for all 53 test dataset.

As discussed above we take each fingerprints from test dataset and find first five fingerprints from training data set which gives least RSS difference then as shown above we find average of location for first three and first five and then found distance between actual location and average location which gives estimation error so here we found error between actual location and avg. of first three location and repeat same things for all available fingerprint so here we have 53 fingerprints so we did for all 53 and then same process we did for finding error distance between actual location and new location which we get from taking average of first five location ,we repeat all this same process for all 53 fingerprints and then we found following result in form of line graph which shown in Figure 8, from this result we can say instead of averaging five co-ordinate if we do average of first three co-ordinate then we can get more accurate location and reduce error distance.

(a) Error distance of Avg. of five fingerprint
From the graph we can say mostly error distance value comes less in three location average compare to five location average, but there are some points where five location average gives less error distance, after observing overall results we can say three location average gives efficient value and less error distance, we did same process of calculation for every fingerprints in JAVA program (Appendix) and we found new location by averaging three closest and five closest fingerprints accordingly and we found Euclidian distance between new measured location and actual location, comparison between these graphs can be mentioned as per following table 1 to show average, median and StdDev here average value
and StdDev values are almost close to each other and value of it is small which indicates that data points are close to each other.

<table>
<thead>
<tr>
<th></th>
<th>Average of first location</th>
<th>Average of first 3 location</th>
<th>Average of first 5 location</th>
<th>Normal Value from closest fingerprints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum error (m)</td>
<td>6.5</td>
<td>8.2</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Minimum error (m)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>StdDev</td>
<td>1.53</td>
<td>1.67</td>
<td>2.76</td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>1</td>
<td>1.2</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>Average Error (m)</td>
<td>1.46</td>
<td>1.79</td>
<td>2.36</td>
<td></td>
</tr>
</tbody>
</table>

Table: 1 Maximum, Minimum, Average values of error distance

Averaging the coordinate of neighbors may give an estimation that is closer to real time location of object then any individual neighbor in our case as shown in figure 9 it gives new location in between all available grids. In our implementation of JAVA program as shown in figure 10 we get output for three location average and new co-ordinate and same for five co-ordinates we get and same way we get for each fingerprints of test dataset. In normal way each fingerprints are compare with all available fingerprints in training dataset ,but in implementation we also provide input console so where we just enter particular fingerprint and measure three location, five location average and Euclidian distance as well normal way Euclidian distance.
Here for more fingerprints accuracy degrades because some points may be far away from true location. Averaging are not useful in small number of locations because nearest neighbors in space are not physically distinct points. In many cases multiple nearest neighbors in space correspond to different orientations at the same point in physical space so averaging in physical space does not improve the location estimation [1].

Conclusion

In this project we deal with test and training datasets and we compare fingerprints of test dataset with each and every fingerprints of training dataset and we found ascending order of fingerprints which are closely match with test dataset’s fingerprint among all those fingerprints we chose first five fingerprints from each comparison of test and training dataset to apply nearest neighbor method. Using nearest neighbor method firstly we take average of three location and then five location and find new location and use this new location to compare with actual location of test dataset, after applying Euclidian distance formula we find distance between new location and actual location, we repeat same process for all 53 dataset which we got from comparison between RSS values of training and test dataset using java program. After comparison and calculation of Euclidian distance we get 53 different value of error distance for k=3 and 5 in nearest neighbor method. After comparing all outputs with each other we can conclude that nearest neighbor method gives more accuracy while averaging three location instead of five location because for five it gives large value of error distance while in three it gives less value of error distance for 41 dataset compare to all 53 dataset. In this way, nearest neighbor method using average of closest three location we can find accurate position of object with less error distance.
INDOOR LOCALIZATION

Reference:

[2] Recent Advances in Wireless Indoor Localization Techniques and System by Zahid Farid, Rosdiadee Nordin, and Mahamod Ismail
[3] An Improved WiFi Indoor Positioning Algorithm by Weighted Fusion Rui Ma *, Qiang Guo, Changzhen Hu and Jingfeng Xue
[6] A Wi-Fi Indoor Localization Strategy Using Particle Swarm Optimization Based Artificial Neural Networks Nan Li,1 Jiabin Chen,1 Yan Yuan,2 Xiaochun Tian,1 Yongqiang Han,1 and Mingzhe Xia1
[7] Fast and Accurate Wi-Fi Localization in Large-Scale Indoor Venues Seokseong Jeon1, Young-Joo Suh1, Chansu Yu2, and Dongsoo Han3
[8] Indoor Positioning Using the Modified Fingerprint Technique Carlos Kornuta1 , Nelson Acosta2 , Juan Toloza3 CONICET, INCA/INTIA - School of Exact Sciences – UNICEN, Tandil, Argentina1 INCA/INTIA - School of Exact Sciences – UNICEN, Tandil, Argentina2 CONICET, INCA/INTIA - School of Exact Sciences – UNICEN, Tandil, Argentina3

Appendix:-

Code for comparing each and every of data set with training dataset and also provide input console to choose particular fingerprint from test data set or compare all fingerprints It gives RSS different between test data set fingerprint and training data set fingerprints in errorData.txt in notepad++. It gives minimum error means minimum RSS difference among all comparison in leastError.txt. It gives Euclidian distance between location of test dataset fingerprint and training data fingerprint, average, maximum, minimum, median variants and standard derivation in EuclidianCalculation.txt

Using nearest neighbor method calculation is done and find closest fingerprint based on RSS difference value and then take average of least three fingerprint location and get new location as well as finding distance between new location and actual location in ThreeEuclidiandata.txt and as per above we did for five fingerprint and get new location found distance between new location and test fingerprint location in FiveEuclidianData.txt. We made following code for implementing our discussed idea.

```java
package optimalerrorfinder;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.Format;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
```
import java.util.HashSet;
import java.util.List;
import java.util.Scanner;

public class OptimalErrFind {
    public static void main(String args[]) throws IOException {
        Scanner sc = new Scanner(System.in);
        System.out.println("Select your choice, Enter a number 1 or 2: ");
        System.out.println("1. Do you want to read entire test File ?");
        System.out.println("2. Do you want to read one line ?");

        String fileName = sc.nextLine();
        BufferedReader test;
        if (fileName.equals("1")) {
            test = new BufferedReader(new FileReader("C:/Users/varil/workspace/Op
timalErrorFinder/src/test.txt");
        } else {
            test = new BufferedReader(new FileReader("C:/Users/varil/workspace/Op
timalErrorFinder/src/testOneLine.txt");
        }
        BufferedReader training = new BufferedReader(new 
FileReader("C:/Users/varil/workspace/Op
timalErrorFinder/src/training.txt");

        ArrayList<String> testFileList = makeTestList(test);
        ArrayList<String> trainingFileList = makeTrainingList(training);

        FileWriter errorDataWriter = (new FileWriter("C:/Users/varil/Desktop/Assignment/errorData.txt");
        FileWriter leastErrorWriter = (new FileWriter("C:/Users/varil/Desktop/Assignment/leastError.txt");

        FileWriter threeEuclidianDataWriter = (new FileWriter 
("C:/Users/varil/Desktop/Assignment/threeEuclidianData.txt");
        FileWriter fiveEuclidianDataWriter = (new FileWriter 
("C:/Users/varil/Desktop/Assignment/fiveEuclidianData.txt");
        FileWriter meanMedianWriter = (new FileWriter("C:/Users/varil/Desktop/Assignment/eucldianCalculations.txt");

        HashMap<Double, String> trainingResultMap = new HashMap<Double, String>();
        HashMap<Double, String> testingResultMap = new HashMap<Double, String>();
        HashMap<Double, String> trainingAllResultMap = new HashMap<Double, String>();
        List<Double> minErrorList = new ArrayList<Double>();
        List<Integer> errorValList = new ArrayList<Integer>();
        HashMap<Integer, List<Integer>> errorTrainingMap = new HashMap<Integer, List<Integer>>();

        for (String testFileLine : testFileList) {
            double totalResult = 0.00;
            double leastResult = 0.00;
            double minError = 5000;
            String testMinErrContributor = ";
            String trainingMinErrContributor = ";
            String testMinErrContributorStr = ";
            String trainingMinErrContributorStr = ";
            errorDataWriter.write("n"+ testFileLine + 
"n");
            String[] testFileLineArr = testFileLine.split("n");
            errorValList.add(Integer.parseInt(testFileLineArr[0]));
        }
    }
}
HashSet<Double> resultList = new HashSet<Double>();

// leastErrorWriter.write(testFileLine + "\n");
for (String trainingLine : trainingFileList) {
    String[] trainingFileLineArr = trainingLine.split("\t");
    String firstTrainingWord = trainingFileLineArr[0];
    if(!firstTrainingWord.equals("-1")) {
        trainingMinErrContributton = trainingLine;
        double result = 0.0;
        for (int testLineWordCount = 1; testLineWordCount <
                testFileLineArr.length;
                testLineWordCount++ ) {
            String testLineWord[] = testFileLineArr[testLineWordCount].split(":");
            String testLineKey = testLineWord[0];
            String testLineValue = testLineWord[1];
            boolean keyExists = false;
            for (int trainingLineWordCount = 1; trainingLineWordCount <
                    trainingFileLineArr.length; trainingLineWordCount++ ) {
                String trainingLineWord[] =
                        trainingFileLineArr[trainingLineWordCount].split(":");
                String trainingLineKey = trainingLineWord[0];
                String trainingLineValue = trainingLineWord[1];
                if(testLineKey.equals(trainingLineKey)) {
                    keyExists = true;
                    int res = (Integer.parseInt(testLineValue) -
                            Integer.parseInt(trainingLineValue));
                    result += Math.pow(res, 2);
                }
            }
            if (!keyExists) {
                int res = Integer.parseInt(testLineValue) + 99;
                result += Math.pow(res, 2);
            }
        }
        double sqrtResult = Math.sqrt(result);
        resultList.add(sqrtResult);
        trainingAllResultMap.put(sqrtResult, trainingLine);
        totalResult = sqrtResult;
        if(totalResult < minError){
            minError = totalResult;
            testMinErrContributor = testFileLine;
            trainingMinErrContributtonStr = trainingMinErrContributton;
            leastResult = totalResult;
            trainingResultMap.put(leastResult, trainingMinErrContributtonStr);
            testingResultMap.put(leastResult, testFileLine);
        }
    }
}
List<Double> resList = new ArrayList<Double>();
for (Double res : resultList) {
    resList.add(res);
}
Collections.sort(resList);
List<Integer> trainingValList = new ArrayList<Integer>();
for (Double res1 : resList) {
    String minTestingError = trainingAllResultMap.get(res1);
    errorDataWriter.write("The minimum error of training file occurred at " + minTestingError + 
"\n");
    errorDataWriter.write("The minimum error of training file occurred at " + res1 + 
"\n");
    System.out.println(res1);
    String[] trainingFileLineArr = minTestingError.split("\t");
    int trainingNodeVal = Integer.parseInt(trainingFileLineArr[0]);
    trainingValList.add(trainingNodeVal);
}
errorTrainingMap.put(Integer.parseInt(testFileLineArr[0]), trainingValList);

System.out.println("The minimum error of testing file occurred at " + testFileLine);
System.out.println("The minimum error of training file occurred at " + trainingResultMap.get(leastResult));
System.out.println("The minimum error is " + minError);
/*
   errorDataWriter.write("The minimum error of testing file occurred at " + testFileLine + 
"\n");
   errorDataWriter.write("The minimum error of training file occurred at " + trainingResultMap.get(leastResult) + 
"\n");
   errorDataWriter.write("The minimum error is " + minError + 
"\n");
*/
minErrorList.add(minError);
Collections.sort(minErrorList);
for (Double minInteger : minErrorList) {
    leastErrorWriter.write("The minimum error is " + minInteger + 
"\n");
    leastErrorWriter.write("The minimum error of testing file occurred at " + testingResultMap.get(minInteger) + 
"\n");
    leastErrorWriter.write("The minimum error of training file occurred at " + trainingResultMap.get(minInteger) + 
"\n");
}
errorDataWriter.close();
leastErrorWriter.close();

BufferedReader corodintes = new BufferedReader(new
FileReader("C:\Users\waril\workspace\OptimalErrorFinder\src\map_coordinates.txt");
ArrayList<String> coordinateFileList = makeCoordinateList(corodintes);

HashMap<Integer, Coordinates> coordinateMap = new HashMap<Integer, Coordinates>();

for (String coordinateVal : coordinateFileList) {
    String[] coordinateValArr = coordinateVal.split(" ");
    String coordinateStr = coordinateValArr[1];
    String[] coordinateArr = coordinateStr.replace("(", ").replace(")", ")").split(",");

    Coordinates coordinate = new Coordinates();
    coordinate.setTestVal(Integer.parseInt(coordinateValArr[0]));
    coordinate.setXaxis(Double.valueOf(coordinateArr[0]));
    coordinate.setYaxis(Double.valueOf(coordinateArr[1]));
    coordinateMap.put(Integer.parseInt(coordinateValArr[0]), coordinate);
    System.out.println("The coordinateList : " + coordinateValArr[0] + ", " + coordinateValArr[1]);
}

for (Integer errorVal : errorValList) {
    System.out.println("The Error value is : " + errorVal);
    System.out.println("The training value is : " + errorTrainingMap.get(errorVal));
double errorXVal = coordinateMap.get(errorVal).getXaxis();
double errorYVal = coordinateMap.get(errorVal).getYaxis();

int i = 0;
int j = 0;
double finalTrainingXVal = 0.0;
double finalTrainingYVal = 0.0;

List<Double> threeXEuclidianList = new ArrayList<Double>();
List<Double> threeYEuclidianList = new ArrayList<Double>();
List<Double> fiveXEuclidianList = new ArrayList<Double>();
List<Double> fiveYEuclidianList = new ArrayList<Double>();
List<Double> threeAverageEuclidianList = new ArrayList<Double>();
List<Double> fiveAverageEuclidianList = new ArrayList<Double>();

double euclidianTotal = 0.0;
List<Double> euclidianList = new ArrayList<Double>();

for (Integer innerTrainingVal : errorTrainingMap.get(errorVal)) {
    double trainingXVal = coordinateMap.get(innerTrainingVal).getXaxis();
    double trainingYVal = coordinateMap.get(innerTrainingVal).getYaxis();
    double euclidianInnerVal = ((errorXVal - trainingXVal)*(errorXVal - trainingXVal))
                               + ((errorYVal - trainingYVal)*(errorYVal - trainingYVal));
    double euclidianFinalval = Math.sqrt(euclidianInnerVal);
    meanMedianWriter.write("Euclidian Value: "+ euclidianFinalval + "\n");
    euclidianList.add(euclidianFinalval);
    euclidianTotal += euclidianFinalval;
    if (i < 5) {
        finalTrainingXVal += trainingXVal;
        finalTrainingYVal += trainingYVal;
        if (i < 3) {
            threeXEuclidianList.add(trainingXVal);
            threeYEuclidianList.add(trainingYVal);
            threeAverageEuclidianList.add(euclidianFinalval);
            System.out.println("euclidian Three Value "+ euclidianFinalval);
            System.out.println("==================================");
        }
        fiveXEuclidianList.add(trainingXVal);
        fiveYEuclidianList.add(trainingYVal);
        fiveAverageEuclidianList.add(euclidianFinalval);
        fiveEuclidianDataWriter.write("Euclidian Five Value "+ euclidianFinalval+ "\n");
    }
    System.out.println("error Value "+ errorVal);
    System.out.println("training value "+ innerTrainingVal);
    System.out.println("error X value "+ errorXVal);
    System.out.println("error Y value "+ errorYVal);
    System.out.println("training X value "+ trainingXVal);
    System.out.println("training y value "+ trainingYVal);
}

System.out.println("euclidian Value "+ euclidianFinalval);
System.out.println("==================================");
// Euclidian Average
double euclidianAverage = euclidianTotal/(errorTrainingMap.get(errorVal).size());
Collections.sort(euclidianList);

// Minimum Value
double minEuclidianVal = euclidianList.get(0);

// Maximum Value
double maxEuclidianVal = euclidianList.get(euclidianList.size()-1);

// Median Value
double median = 0.0;
int middle = euclidianList.size()/2;
if (euclidianList.size()%2 == 1) {
    median = euclidianList.get(middle);
} else {
    median = (euclidianList.get(middle - 1) + euclidianList.get(middle))/2.0;
}

// Variance Value
double mean = euclidianTotal/euclidianList.size();
double varianceTotal = 0.0;
for (Double euclidianVal : euclidianList) {
    varianceTotal += Math.pow((euclidianVal - mean), 2);
}
double varianceVal = varianceTotal / euclidianList.size();

double threeAvgXVal = 0.0;
for (Double threeXVal : threeXEuclidianList) {
    threeAvgXVal += threeXVal;
}
double finalThreeXTrainingVal = threeAvgXVal/3;

double threeAvgYVal = 0.0;
for (Double threeYVal : threeYEuclidianList) {
    threeAvgYVal += threeYVal;
}
double finalThreeYTrainingVal = threeAvgYVal/3;

double euclidianThreeInnerVal = (((errorXVal - finalThreeXTrainingVal)*(errorXVal - finalThreeXTrainingVal)) + ((errorYVal - finalThreeYTrainingVal)*(errorYVal - finalThreeYTrainingVal)));
double euclidianThreeFinalval = Math.sqrt(euclidianThreeInnerVal);

meanMedianWriter.write("Euclidian Average: " + euclidianAverage + 
meanMedianWriter.write("Euclidian Minimum Value: " + minEuclidianVal + 
meanMedianWriter.write("Euclidian Maximum Value: " + maxEuclidianVal + 
meanMedianWriter.write("Euclidian Median: " + median + 
meanMedianWriter.write("Euclidian Variance: " + varianceVal + 
meanMedianWriter.write("Euclidian Standard Deviation: " + standardDeviation + 

meanMedianWriter.write("Euclidian Three Value: " + euclidianThreeFinalval);
System.out.println("Euclidian Three Value: ");
System.out.println("=====================================");
threeEuclidianDataWriter.write("Avg of min 3 coordinate = (" + finalThreeXTrainingVal + ", " + finalThreeYTrainingVal + ")\n");
threeEuclidianDataWriter.write("Euclidian Three Average Value " + euclidianThreeFinalval+ 
"\n\n");

double fiveAvgXVal = 0.0;
for (Double fiveXVal : fiveXEuclidianList) {
    fiveAvgXVal += fiveXVal;
}
double finalFiveXTrainingVal = fiveAvgXVal/5;

double fiveAvgYVal = 0.0;
for (Double fiveYVal : fiveYEuclidianList) {
    fiveAvgYVal += fiveYVal;
}
double finalFiveYTrainingVal = fiveAvgYVal/5;

double euclidianFiveInnerVal = (((errorXVal- finalFiveXTrainingVal)*(errorXVal- finalFiveXTrainingVal)) + ((errorYVal- finalFiveYTrainingVal)*(errorYVal- finalFiveYTrainingVal)));
double euclidianFiveFinalval = Math.sqrt(euclidianFiveInnerVal);

System.out.println("euclidian Five Value " + euclidianFiveFinalval);
System.out.println(" Euclidian Five Average Value " + euclidianFiveFinalval+ "\n\n");
}
fiveEuclidianDataWriter.write("Avg of min 5 coordinate = (" + finalFiveXTrainingVal + ", " + finalFiveYTrainingVal + ")\n");
fiveEuclidianDataWriter.write("Euclidian Five Average Value " + euclidianFiveFinalval+ "\n\n");
}

private static ArrayList<String> makeTrainingList(BufferedReader training) throws IOException {
    ArrayList<String> trainingList = new ArrayList<String>();
    String trainingData;
    while((trainingData = training.readLine())!=null){
        trainingList.add(trainingData);
    }
    return trainingList;
}

private static ArrayList<String> makeTestList(BufferedReader test) throws IOException {
    ArrayList<String> testList = new ArrayList<String>();
    String testData;
    while((testData = test.readLine())!= null){
        testList.add(testData);
    }
    return testList;
}

private static ArrayList<String> makeCoordinateList(BufferedReader test) throws IOException {
    ArrayList<String> coordinateList = new ArrayList<String>();
    String testData;
while((testData = test.readLine())!= null){
    coordinateList.add(testData);
}
return coordinateList;
}