ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

Patrick Elsener

Towards High Data Quality in Mobile Crowdsensing

Bachelor Thesis (BA)-2017-18
October 2017 to March 2018

Tutor: Balz Maag
Co-Tutor: Zimu Zhou
Supervisor: Prof. Dr. Lothar Thiele

Acknowledgments

I would like to thank my advisors, Balz Maag and Dr. Zimu Zhou, for mentoring this thesis. They always knew how to assist me with their experience in academic projects, as well as providing me with a lot of information about the ongoing research in the area of mobile crowdsensing. Another big thank goes to the entire Computer Engineering Group at TIK under the lead of Prof. Dr. Lothar Thiele for giving me the opportunity to work on this thesis and providing me with the necessary equipment.

Abstract

With the widely used smartphones and growth in popularity of other wearable devices such as smartwatches, nearly every user becomes a possible participant in a mobile crowdsensing project. As these possible participants are no experts in taking measurements, every data point has a context attached to it, defined by the activity of the participant during the measurement. In this thesis we conduct our own mobile crowdsensing measurements with different environmental sensors in various popular user contexts. With the results of the measurements we can observe that there exists a relationship between context and data quality. We test a variety of learning algorithms to estimate the ground truth from our measurement data. The results of our evaluation are mixed. While some algorithms are able to remove the influence of the context on the data others are not suited for this task. We also test a truth finding algorithm, which turns out to not be able to recognize the relationship between context an data quality.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

1.1 Motivation

Nowadays smartphones are widely used and other wearable devices (i.e. smartwatches) get more and more popular. All these devices are shipped with a big selection of sensors equipped. This makes nearly everyone able to collect sensor data and therefore possible participants in crowdsourcing projects [7]. Participants of such projects are by no means experts in collecting data, which leads to the question how good the quality of the collected data really is. One can imagine that not all participants behave the same way as they collect the data. A collected data point has therefore a context attached to it, given by the activity of the participant during the measurement. It is, to our knowledge, currently little known how these contexts influence the quality of crowedsourced data.
In this thesis we want to investigate the potential influence of different contexts on data quality and how we can possibly invalidate this influence. To get data from mobile sensors in specific contexts, we conduct our own measurements which we perform in different contexts. From the collected data we try to answer the question how the contexts influence the data quality. In a last step we develop and evaluate algorithms to invalidate the effects of the contexts on the data.

1.2 Related Work

We consider the following listed papers as related to this thesis. They either served us as an inspiration and motivation or they helped us to build part of this thesis based on their results.

Context-Aware Data Quality Estimation in Mobile Crowdsensing The authors in [1] are one of the first to consider the influence of context on data quality in mobile crowdsensing. They were able to improve the data quality by developing an algorithm which takes the contexts attached to the data into account. They used their algorithm to guide user recruitment in mobile crowdsensing in an online manner.
Their observation of context corresponding to data quality builds the basis for this thesis. We want to improve this paper in two points. First we want to consider more than just temperature data and secondly we want to consider the context in our data from the beginning to the end of our ground truth estimation.

A Survey on Truth Discovery The work in [2] serves as a great overview of different truth discovery methods. Each of these methods is summarized from different aspects and for each of them it is listed on which kind of data sets they are usable.
This paper helps us to pick a suitable truth discovery method for our data set. Then we test this method on how it can deal with different contexts. This method also serves us as a comparison to our own method to tackle the problem of contexts having an influence on data quality.

Participatory Sensing or Participatory Nonsense? - Mitigating the Effect of Human Error on Data Quality in Citizen Science The work in [3] sees crowdsensing as an alternative to

traditionally conducted measurements by trained personnel. The authors of this paper worry about the errors in the data due to measurements performed by non-experts as it is usual for crowdsensing projects. In the first part of the paper the authors did a study on the errors and behaviors of non-experts performing measurements. They were able to observe a huge variety of different behaviors and errors. In the second part of the paper they tested how technical measures and instructions can prevent unwanted behavior and errors during measurements.
The first part of this paper serves us as a proof for that every data point of a crowdsensing project has a context attached to it which could potentially lead to errors in the data.

Complex Human Activity Recognition Using Smartphone and Wrist-Worn Motion Sensors The authors in [4] show that it is possible to accurately recognize an activity with a smartphone in the pocket and with additional motion sensors worn at the wrist in a smartwatch like fashion. They are also able to show that the combination of the smartphone together with the additional motion sensors at the wrist outperforms the activity recognition of using only the smartphone or the motion sensors at the wrist exclusively.
This paper allows us to assume that activity recognition is a solved problem. Therefore, we can work with different activities/contexts without worrying how to recognize them by ourself.

1.3 Report Structure

In chapter 2 we describe our own mobile crowdsensing measurements. In section 2.1 we explain different decisions regarding the hardware we use, the context and sensor types we consider and what software we use. Section 2.2 describes in detail how the measurements were performed. In section 2.3 we analyze the data from the measurements and investigate the relationship between context and data quality.
In chapter 3 we describe how we tackle the problem of ground truth estimation. First we explain what the goal of ground truth estimation is (section 3.1). Then we describe the different algorithms we use for the ground truth estimation, what their input is and how we score their performance (section 3.2). In the last part of this chapter we introduce a truth finding algorithm as comparison to the ground truth estimation (section 3.3).
Chapter 4 evaluates the results of chapter 3. First we discuss the performance of the different algorithms used for the ground truth estimation in section 4.1. The performance of the truth finding algorithm is then discussed in section 4.2 as well as how it deals with the different contexts.
Lastly, in chapter 5 we conclude this thesis. We summarize the findings of this thesis in section 5.1. In section 5.2 we discuss what and how a possible future work can improve this thesis. Section 5.3 is dedicated to take-away messages for someone handling data coming from a mobile crowdsensing context.

Chapter 2

Measurements and Observations

In this Chapter we discuss how we set up our own mobile crowdsensing measurements. In section 2.1.1 we first describe which context we have chosen and why. Section 2.1.2 describes the hardware we used and why we have decided us for it. Section 2.1.3 lists the sensor types we focus on in this thesis. In Section 2.1.4 we describe the complementary software we used and how we had to adapt existing ones.
In Section 2.2 we describe our mobile crowdsensing approach. We first describe, in section 2.2.1, how we compared the multiple sensor board with each other. Section 2.2.2 goes into details how we performed the mobile crowdsensing measurements in context.
In section 2.3 we list our observations from the two measurements described in section 2.2.

2.1 Measurement Tools

2.1.1 Contexts

One can think of a lot of possible contexts in which a person can take a measurement. We surely can not consider all of them. Instead we lay focus on a selection of contexts.
We have chosen standing still, walking, running and office work as our contexts. The contexts are explained in table 2.1. The chosen contexts are proven to be recognizable by motion sensors in [4].
We further decided to distinguish whether a measurement is performed outdoors or indoors. Outdoor and indoor provide two totally different environments, which can have different influences on the measurements. The contexts standing still and walking are used outdoors as well as indoors. Running is only used outdoors, because we think it is rather unusual having a person conduct a measurement indoors while running. A person who is running or jogging outdoors and taking some measurements is more likely. For the office work context it is quite similar. Usually a person is doing some office work within a building, therefore we use the context office work only indoors.

2.1.2 Hardware

We were looking for a sensor board or sensor platform that resembles typical wearable devices that include a wide range of sensors such as the current apple watch [5] or an advanced fitness tracker [6]. The hardware solution has also to support sensors commonly used in crowdsensing projects.
As the hardware resembles typical wearable devices, the price is also a factor to consider. Wearable devices cost very little up to a moderate price. This factor has to be reflected in our hardware choice as well.
Currently a big price span can be observed when looking at the different air quality sensors. The professional air quality sensors are priced in the thousands of US dollars. These professional air quality sensors are also meant to be installed stationary and not to be carried around. Luckily for us, there exist some affordable air quality sensors which are also considerably smaller. The

Context	Description	Indoor/Outdoor	Activities
Standing Still	Low physical activity with nearly no movement of the arms.	Indoor and outdoor	Standing still, sitting still with no movement of the arms.
Walking	Walking with normal speed, medium physical activity, natural movement of the arms.	Indoor and outdoor	Walking around, using smaller stairs.
Running	Any form of running, high physical activity.	Outdoor	Running and jogging.
Office Work	Every kind of activity that is possible at an office desk.	Indoor	Typing, sorting papers, opening letters or having a drink.

Table 2.1: Context description with example activities and used environments

big price span indicates that the quality of the affordable air quality sensors is unfortunately not comparable to the good quality of a cheap temperature sensor.

Thunderboard We use the Thunderboard from Silicon Labs, a small sensor board with a large variety of sensors on it. The available sensors range from temperature, humidity, pressure, ambient light and UV to motion, acceleration, sound level and air quality (Figure 2.1). The Thunderboard features sensors used in crowdsensing projects such as Noisetube [7] or OpenSense 2 [8].
The sensor board features a micro USB connection through which the board can be powered. It is also possible to run the Thunderboard in battery mode via a coin cell battery. In this mode the air quality sensor is disabled due to its high energy consumption.
There are two possible ways to read out the sensor values. One way is via the official Thunderboard app (for iOS and Android) connected via Bluetooth. The other way is via the USB port. The standard output of the firmware is linked to the serial port and the sensor values can be written to it by altering the firmware running on the sensor board. Silicon Labs provides the source code of the firmware within their development environment Simplicity Studio [9].

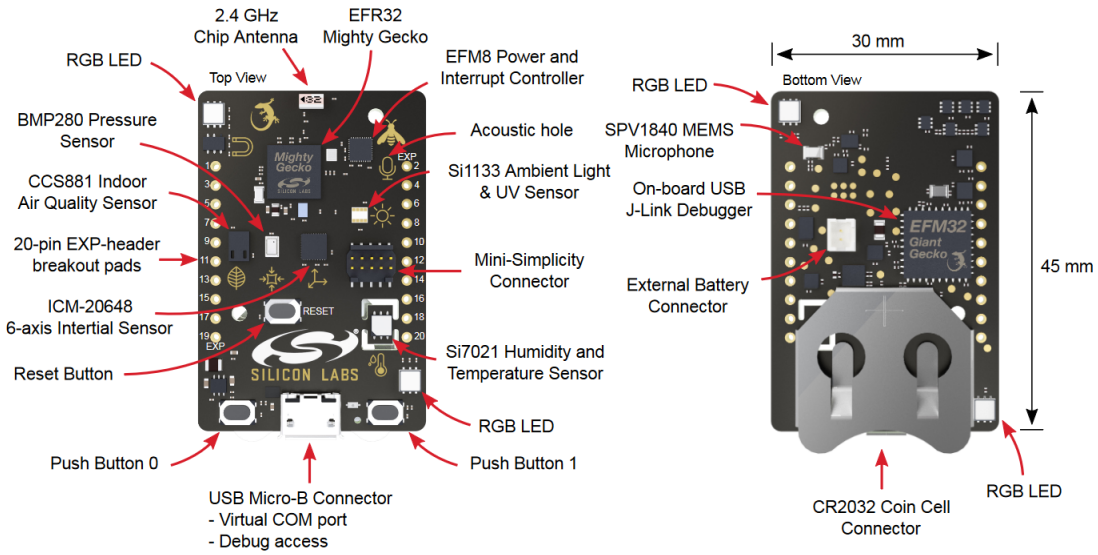

Figure 2.1: Front and back view of a Thunderboard from Silicon Labs

Android Phone We use a Samsung Galaxy Note 4 running Android 6.0.1 to connect to the sensor board via the official Thunderboard app.
This combination of sensor board and phone allows us to have the Thunderboard carried around by a person and extracting sensor values from it at the same time. The sensor data from the Thunderboard is stored on the phone's internal memory and is later transfered to a computer for further evaluations.

2.1.3 Sensors

Based on our hardware decision we focus on five sensor metrics. These five metrics are temperature, humidity, sound level, CO_2 concentration and the concentration of total volatile organic compounds (TVOC). CO_2 and TVOC are used to determine the indoor air quality.
The five metrics provide a good indication of the surrounding environment's quality.

2.1.4 Software

With just the hardware it is not possible to store the sensor values from the Thunderboard. Therefore, the use and modification of additional software is necessary.

Thunderboard App Silicon Labs provides the source code of the Thunderboard app for both the Android version as well as for the iOS version [10]. We downloaded the Android version and added a new section to it. In this new section it is possible to start a measurement for a specific context (Figure 2.2).
When a measurement is started, the app logs all sensor values from the Thunderboard and stores them in a csv file on the phone.

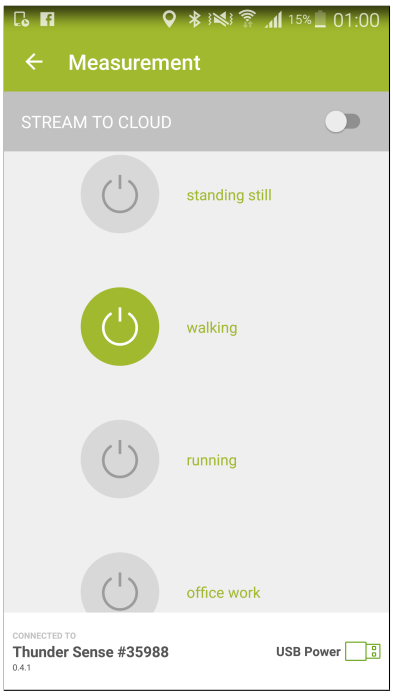

Figure 2.2: Added section to the Android app with running measurement in walking context

Thunderboard Firmware To be able to read the sensor values directly from the Thunderboard's serial port, we modified its firmware. Silicon Labs provides the source code for the firmware via their development environment Simplicity Studio [9].
The modified firmware collects the sensor values and writes them to the standard output, which

is linked to the serial port. A further modification prevents the Thunderboard to enter power safe mode, if no Bluethooth connection is established.

Script for Reading from the Serial Port To read the sensor values from the Thunderboar's serial port and storing them on a computer, we had to write a small Python script. The script connects to a serial port of a computer and filters out the sensor values from the Thunderboard's standard output. The sensor values are stored in a csv file on the computer.

2.2 Measurement Approach

2.2.1 Overnight Measurement

The goal of the first measurement is to compare three Thunderboards to each other. This allows us to see if their measurements correspond to each other or if there are differences between them. If the quality of two sensors is too bad, their values would differ widely from each other. If values of a sensor always show a constant offset, the sensor itself is not calibrated correctly.
For this measurement, we connected three Thunderboards to a computer and placed them next to each other on a desk located indoors. All three Thunderboards collected data over night in a two second interval for about nine and a half hours.

2.2.2 Measurements in Context

The main reason why we do our own measurements is to get sensor values which where taken in a specific context.
Two Thunderboards are needed per measurement. One serves as a provider for the ground truths (we assume these values to be the true ones). This Thunderboard is connected to a notebook and is static during the measurement, hence we also refer to it as static sensor. The other sensor board is worn by a person at the wrist position, like a smart watch. This person also carries around the Android phone to which the Thunderboard is connected to. We refer to this Thunderboard as mobile sensor. During a measurement, the person acts according to the context the measurement is desired to be in (i.e. walking). Both Thunderboards log their data in a frequency of two seconds and each measurement is around two minutes long.
Rather to take long measurements we keep them short but perform multiple of them. The idea behind this is to get more variance in our data, otherwise it is possible that one moment (= one measurement) is represented strongly in our data.
The outdoor measurements were performed during the late fall (mid to end of November) which means that they were performed during rather cold weather with very little sunshine.
Some sensors need a startup time before they give legit values. To prevent having corrupted values in our data we remove the ten first sensor value of each measurement. We further smooth our data by aggregating it. We define a window with a width of ten sensor values. All sensor values within the window get averaged (average of temperature values, average of humidity values, etc). The window slides over the data by a step size of five sensor values, which results in a window overlap of 50%. Instead of only using the average in the window we also use the median. We are later able to see if one is superior over the other (section 2.3.2).

2.3 Observations

2.3.1 Observations from the Overnight Measurement

Looking at the sensor values of the three Thunderboards we ran over night, we see that they show similar values to each other. As seen from the sensor value graphs (Figures 2.3, 2.4, 2.5, 2.6) the curves of the values all follow the same overall path. We also see that the sensor values are not exactly the same. The difference is more or less a constant offset. One Thunderboard for example has always a lower temperature value than the other two Thunderboards but the curve of this sensor moves in a similar way to the other two. This constant offset indicates a bad calibration of the different sensors on the Thunderboards. Overall the measurements of the

three Thunderboards are in sufficient agreement for our purposes.
Worth mentioning is the fact that the air quality sensor has much more noise in its data. They do not always have a constant offset to each other. Further inspection shows that the air quality sensor reacts very sensitive but also very slowly. Touching the air quality sensor gives a huge peak in the data. It also takes some time until the data is back to normal. As already suspected in the hardware section (2.1.2), this indicates that the quality of the air quality sensor is not as desired.

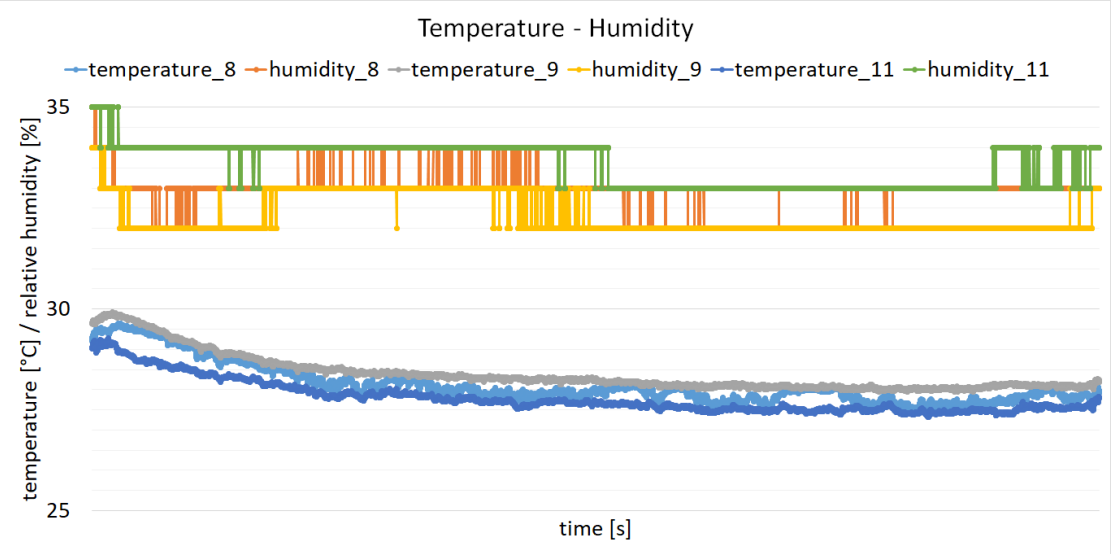

Figure 2.3: Overnight measurement of three Thunderboards, temperature and humidity values, The numbers 8, 9 and 11 correspond to the port to which the Thunderboards connect to

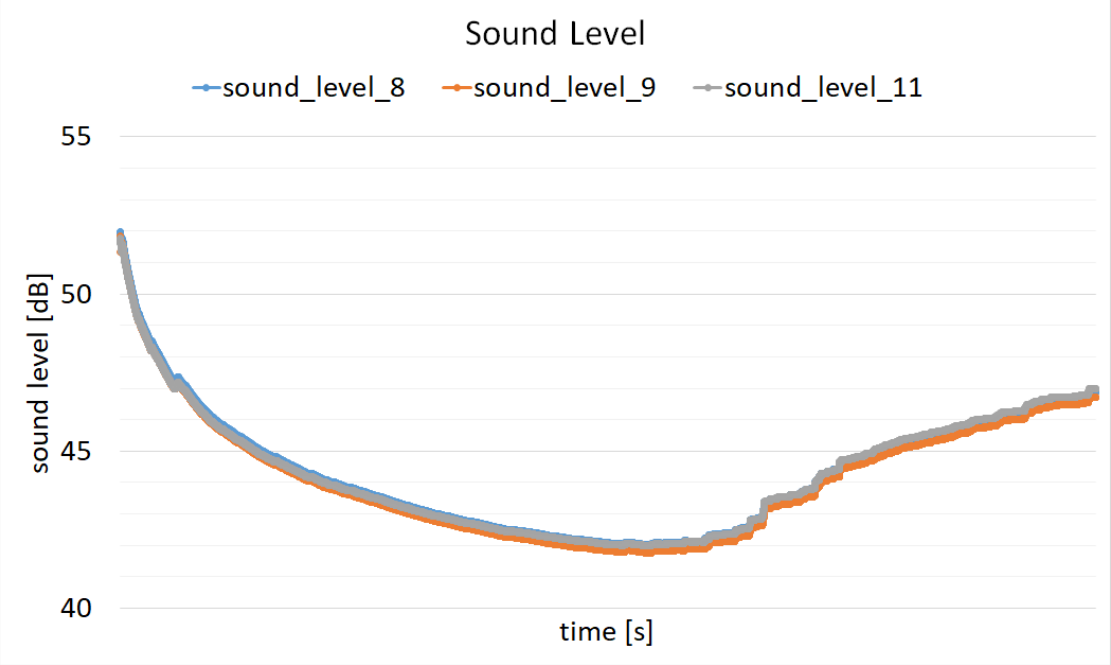

Figure 2.4: Overnight measurement of three Thunderboards, sound level values,
The numbers 8, 9 and 11 correspond to the port to which the Thunderboards connect to

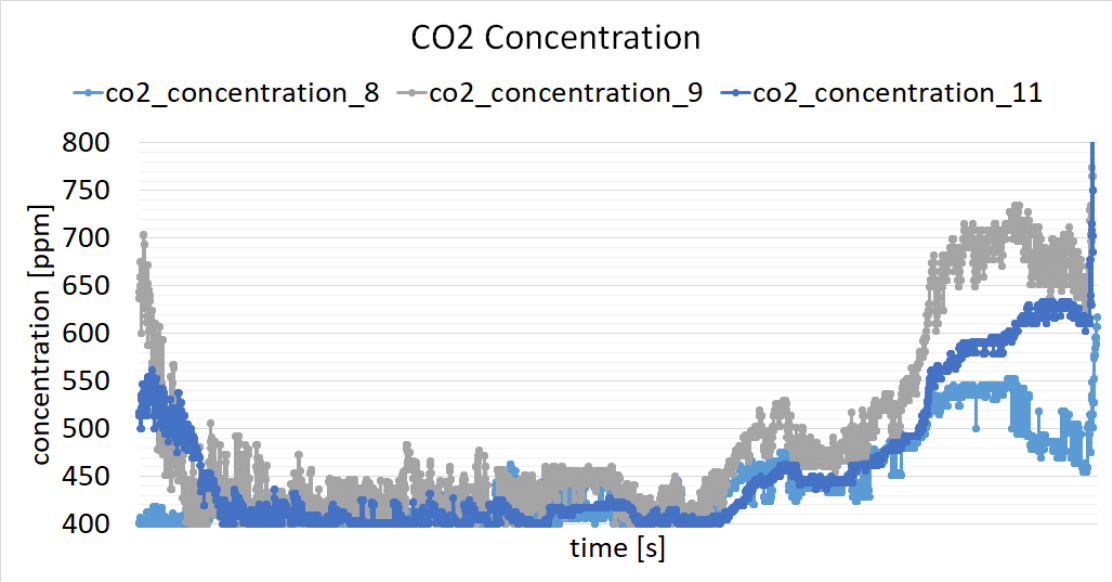

Figure 2.5: Overnight measurement of three Thunderboards, CO_2 concentration values, The numbers 8, 9 and 11 correspond to the port to which the Thunderboards connect to

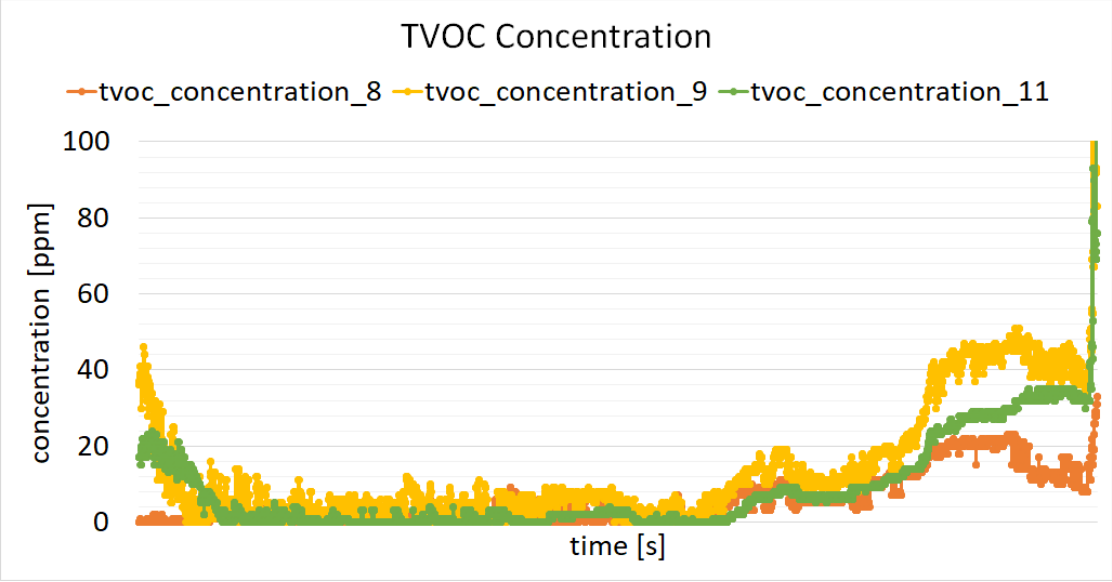

Figure 2.6: Overnight measurement of three Thunderboards, TVOC concentration values, The numbers 8, 9 and 11 correspond to the port to which the Thunderboards connect to

2.3.2 Observations from the Measurements in Context

We can see that the different contexts have an influence on the quality of the different sensor values. The context is not the only influence on the data quality. There is also a visible difference in data quality between the outdoor and indoor measurements (Table 2.2).

	Temperature			Humidity			Sound level			CO_2 concentration			TVOC concentration		
	mean	median	std. dev.	mean	median	std. dev.	mean	median	std. dev.	mean	median	std. dev.	mean	median	std. dev.
Total	0.233	0.220	0.176	-0.169	-0.196	0.091	0.005	-0.003	0.125	1.091	0.628	1.562	61.493	2.470	270.475
Indoor	0.167	0.215	0.089	-0.170	-0.219	0.099	-0.007	-0.012	0.138	1.582	1.259	1.780	91.645	3.183	336.387
Outdoor	0.344	0.321	0.224	-0.168	-0.171	0.077	0.025	0.010	0.098	0.255	0.212	0.294	10.112	2.099	29.290
Standing still combined	0.332	0.199	0.249	-0.142	-0.144	0.087	-0.079	-0.057	0.109	0.754	0.096	1.479	39.465	1.000	194.092
Standing still indoor	0.145	0.150	0.048	-0.087	-0.088	0.069	-0.127	-0.087	0.130	1.507	0.757	1.866	76.154	1.000	276.543
Standing still outdoor	0.502	0.495	0.234	-0.192	-0.190	0.070	-0.036	-0.033	0.059	0.076	0.062	0.232	6.457	1.000	24.546
Walking combined	0.151	0.081	0.189	-0.112	-0.101	0.102	-0.017	-0.003	0.080	1.754	0.463	2.444	150.270	4.483	469.015
Walking indoor	0.025	0.002	0.065	-0.048	-0.033	0.077	-0.022	0.002	0.103	3.077	2.774	2.851	281.430	3.950	630.033
Walking outdoor	0.280	0.277	0.187	-0.179	-0.198	0.081	-0.012	-0.009	0.045	0.386	0.318	0.264	14.663	4.646	37.935
Office work indoor	0.224	0.222	0.023	-0.237	-0.241	0.031	0.032	0.008	0.129	1.080	1.266	0.564	29.723	9.046	91.519
Running outdoor	0.247	0.277	0.150	-0.131	-0.147	0.066	0.131	0.110	0.089	0.296	0.260	0.292	8.887	2.038	20.887

Table 2.2: Statistics of the relative errors from mobile crowedsensing measurements

Mean vs. Median Calculating the relative errors (equation 2.1) between the mobile and the static sensor data by either the difference of the means (window aggregation via mean) or by the difference of the medians (window aggregation via median) makes little difference to the temperature, humidity, sound level and CO_2 concentration values. The total volatile organic compound concentration shows a different picture. Using the median leads to much smaller relative errors. This is not expected as the CO_2 concentration and the TVOC concentration is calculated on the sensor from the same measured value. One possible explanation is that the peaks occurring in the data are so big (value wise, not quantity wise) that they have a big influence on the mean, where the median is known to be more robust against such outliers.

$$relative\,error = \frac{estimate - truth}{truth} \tag{2.1}$$

Indoor vs. Outdoor Comparing indoor and outdoor data from the walking and standing still contexts show for temperature and humidity values that the errors from outdoor data are higher compared to the indoor data, the standard deviations of the outdoor data are also higher. This is most likely due to the higher temperature difference between the outdoor temperature and the skin temperature and due to the higher difference between the outdoor humidity and the skin environment humidity respectively. Measurements were made mid to end of November, which means that the temperature and humidity outside was relatively low. The relative errors and the standard deviation of the sound level measurements show exactly the opposite picture. It looks like the higher background noise outdoors covers a big portion of the noise produced by the person wearing the sensor board during the measurements. The two air quality values both perform better in outdoor conditions (both relative errors and standard deviations are smaller). One must keep in mind that the air quality sensor was meant to be used indoors. We assume that the two air quality metrics are heavily influenced by urban pollutants such as ozone.

Temperature and Humidity In both, indoor and outdoor conditions, some movement from the person conducting the measurement seems to be beneficial for the temperature and the humidity sensor values considering the relative errors, the standard deviations show similar values. The biggest difference is seen when comparing standing still and walking. Running is even better than walking but just slightly, considering both relative errors and standard deviations. Office work performs the worst, but it has the smallest standard deviations. In conclusion one can say that a certain airflow is beneficial to the temperature and humidity sensor.

Sound Level The sound level values shows similar results to the temperature and humidity sensors when looking at standing still and walking contexts. The running context performs the worst. This is probably due to the captured wind and the noise produced by the person wearing the Thunderboard. Surprisingly, the office work context performs better than the standing still context, despite the mobile sensor being closer to potential noise sources.

CO_2 Concentration and TVOC Concentration The sensor values from the walking contexts have worse relative errors and standard deviations than those from the standing still context. More movement in form of the running context gives better results than the sensor values from the walking context, but still worse than those from the standing still context. It seems that there is a trade off between physical activity and airflow. Considering the office work context, the two sensor values perform even better than in the standing still context, with the lowest standard deviations of all four contexts. Despite the mobile sensor being in breath range it does not seem to be affected by this. The good performance in this context is potentially caused by the small movement of the mobile sensor, which seems to have a beneficial influence on the air quality sensor.

Combined Data Looking at the influences the different contexts have on the different sensor values as well as considering the above discussion of indoor versus outdoor, it clearly makes sense to separate the data by context and the environment (indoor or outdoor). Combined data, such as walking combined (walking indoor and walking outdoor together), have in general a

bigger standard deviation as the individual contexts itself. The same applies for the combined indoor data and the combined outdoor data.

Chapter 3

Ground Truth Estimation

In section 3.1 we explain why we are interested in ground truth estimation. The different algorithms which we use to estimate the ground truth are described in section 3.2.1. How we handle the input for these algorithms is described in section 3.2.2. To make the different algorithms comparable to each other we have to score them, which is explained in section 3.2.3.
In section 3.3 we explain why we also use a truth finding algorithm. The specific algorithm we use is further described in section 3.3.1 and in section 3.3.2 we explain how we create the input data for this algorithm.

3.1 Goal of Ground Truth Estimation

We want to estimate the ground truth from the data produced by the measurements performed in a specific context (Section 2.2.2). We exploit the potential relationship between context and measurement quality to recover the ground truth of the measurement data. We assume the value provided by the static sensor as the ground truth. We decided to try a variety of common machine learning algorithms as well as two simple neural networks. The goal is to analyze their performance afterwards to see if one of them is able to correct the errors introduce by the context in an acceptable manner.

3.2 The different Ground Truth Estimation Algorithms

3.2.1 Short Description of the Algorithms

Ridge Regression and Bayesian Regression We use two variants of Ridge Regression. One with only a one dimensional input, which uses only the corresponding sensor value to estimate a certain sensor value ground truth (i.e. using only the temperature value to estimate the temperature ground truth). The other variant considers all five sensor values to estimate the ground truth. On both variants we use cross validation to determine the gamma parameter, which penalizes learned weights of high magnitudes and, therefore, forces the learned weights to have a small magnitude.
Bayesian Regression is comparable to Ridge Regression but the gamma parameter is learned within its mathematical model.
For both, Ridge Regression and Bayesian Regression, we use the implementation provided by scikit-learn [11] [12].

Bayesian Regression with Data transformed into Polynomial Space of Degree 2 and 3 We use Bayesian Regression another two times, but with transformed input data. We transform our data once in a polynomial space of degree two and once in a polynomial space of degree three.
For the Bayesian Regression as well as for the data transformation we use the implementations provided by scikit-learn [12] [13].

Kernel Ridge Regression with Polynomial Kernel of Degree 2 Instead of transforming the data into a polynomial space as described above (3.2.1), an alternative is to make use of the kernel trick. To archive this we use Ridge Regression with a polynomial kernel of degree two.
We used the implementation provided by scikit-learn [14].

SVR with RBF, Sigmoid, Linear Kernel and Polynomial Kernel of Degree 2 and 3 The kernel trick we use for the Ridge Regression is also applicable to Support Vector Regressions. We use five different kernels: a RBF kernel, a sigmoid kernel, a linear kernel and two polynomial kernels of degree two and three respectively.
We use the implementation provided by scikit-learn [15].

KNN with Uniform and Distance Weights To complement the above algorithms, which focus on fitting a mathematical model on the data, we include also an algorithm which focuses on the distance of the data points to each other.
We use two varieties of the K Nearest Neighbor algorithm. One variety uses uniform weights, which means that every data point in the neighborhood contributes the same to the cluster assignment of a query point. The other variety uses distance weights which give data points closer to the query point more weight. For both varieties the K was either three, five or seven, determined by cross validation.
For both varieties we use the implementation provided by scikit-learn [16]

Multi-Layer Perceptron Neural Network To round up our collection of algorithms we also include two simple neural networks. For the first neural network we use scikit-learn's implementation [17]. The neural network has two hidden layers with the first hidden layer having 128 neurons and the second layer having 64 neurons. As optimizer we use lbfgs, which is an optimizer in the family of quasi-Newton methods. The second neural network is built with tensorflow [18]. This neural network has again two hidden layers but has less neurons, 64 and 16 neurons respectively. In addition, the second neural network has a dropout layer on top for regularization reasons. As optimizer we use the momentum optimizer which is provided within tensorflow.

Baseline Algorithms We need a basis for the performance of the algorithms described above. For this reason we create two baseline algorithms. The first one calculates the score 3.2.3 on the data without any modifications. For the second control algorithm we use a simple method which corrects the data by the median offset to the ground truth.

3.2.2 Data Input for the Algorithms

For each sensor type (temperature, humidity, sound level, CO_2 and TVOC concentration) there are eleven data sets as input. These data sets are total (all measurements combined), indoor (all measurements coming from an indoor environment combined), outdoor (all measurements from an outdoor environment combined), standing still indoor, standing still outdoor, standing still (combination of the data from the previous two data sets), walking indoor, walking outdoor, walking (combination of the data from the previous two data sets), running outdoor and office work.
Each of the data sets are randomly split into a training and test set. After the split the training set goes through a balancing process. This is done by assigning the data points (regarding the ground truth) to uniformly (between the maximum and minimum of the data set) distributed bins. From each of these bins a certain number of data points is taken, defined by the formula 3.1.
Some algorithms benefit or even require the input data to be centered. If this is the case for an algorithm, we center its input data.

$$\text{take from each bin} = \begin{cases} \left\lfloor \frac{n}{\#ofbins} \right\rfloor, & \text{if } > 0 \\ 5, & \text{otherwise} \end{cases}, n := \text{\# of data points} \tag{3.1}$$

3.2.3 Scoring the Algorithms

To score the algorithms we use the root mean square error (RMSE). The RMSE score is defined by the equation 3.2 [19]. We choose the RMSE over the mean absolute error (MAE) because RMSE is sensitive to outliers. RMSE is sensitive to outliers because the errors get squared before averaged. MAE, on the other hand, behaves proportionally to the error magnitude because the absolute value of the errors is taken before averaging. We especially want to correct outliers with the algorithms and, therefore, we want to punish an algorithm which leaves outliers in the data.

$$\text{RMSE} := \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \quad N = \text{\# of samples}, y_i = \text{observed value}, \hat{y}_i = \text{predicted value} \tag{3.2}$$

We run each algorithm ten times and take the average of the RMSE scores as final score. To make it fair for every algorithm, we run them with the same inputs respective to the different runs. So algorithm A has the same input for its first run as algorithm B has on its first run. Obviously, the input for the first run is different from the input of the second run and so on.

3.3 Truth Finding Algorithm as Comparison

We are not the first trying to find the ground truth in some data set. There exists a category of algorithms, called truth finding algorithms, especially designed to find the ground truth in data coming from different sources. We test one of these algorithms to see how it performs in comparison to our own approach.

3.3.1 Description of the Truth Finding Algorithm GTM

We use a truth finding algorithm called Gaussian Truth Model (GTM) [20]. GTM uses a Gaussian distribution for each truth to model the probabilistic distribution of observing each claim. The truth itself is used as mean of the Gaussian distribution. The variance parameter controls how likely claims deviate from the truth. Sources which make claims that deviate more from the truth correspond to a larger variance. With this observation GTM models also every source with a Gaussian distribution, which models how likely a claim of this source is. The variance parameter of the Gaussian distribution is used to reflect the quality of the source. The model of the truth and the models for the different sources form an EM model that builds the basis of the GTM algorithm. GTM gives two outputs, one being the found truths and the other one being the assigned quality levels of the sources.

3.3.2 Input Transformation for GTM

Like many other truth finding algorithms GTM expects claims for a truth coming from different sources. Therefore, we have to create such sources. Our sources are based on the different contexts described in section 2.1.1. We define two categories of sources, one for all indoor sources and one for all outdoor sources. For a specific run we use only sources from one category, because it makes no sense having an indoor source and an outdoor source making a claim for the same truth.
Once the different number of sources for one run of the GTM algorithm are determined and the corresponding claims are associated, the input data has to be z-scored (zero mean and standard deviation of one). GTM requires this step because otherwise one source could dominate the other sources just because of the scale of its claims.
It is possible to give GTM the real truths as input. This helps GTM with its EM approach to start close to the solution. We make use of this option because our input is on the smaller side and an initial guess helps an iterative algorithm like GTM.

Chapter 4

Evaluation

In this chapter we discuss the results of the ground truth estimation and the result of the truth finding algorithm from chapter 3. In section 4.1 we discuss the result of the ground truth estimation algorithms, described previously in section 3.2.1. Section 4.2.1 analyzes the performance of the truth finding algorithm, GTM, described in section 3.3.1. A closer look at how GTM handles the different contexts is given in section 4.2.2. Section 4.3 concludes this chapter by listing the key findings of the evaluation.

temperature	total	indoor	outdoor	standing still	standing still indoor	standing still outdoor	walking	walking indoor	walking outdoor	office work indoor	running outdoor
no learning score	4.388	4.756	3.763	4.612	3.869	5.170	2.414	1.772	3.064	5.635	2.420
median shift	2.252	2.252	1.966	1.599	1.160	1.630	1.855	1.672	1.583	0.543	1.246
ridge regression single input	2.100	1.021	1.906	1.542	1.111	1.577	1.683	1.281	1.559	0.458	1.120
ridge regression	1.791	0.873	1.796	1.431	0.889	1.370	1.268	0.820	1.383	0.456	1.229
bayesian regression	1.791	0.887	1.794	1.404	0.884	1.387	1.268	0.797	1.383	0.447	1.210
bayesian regression poly d2	1.583	0.876	1.637	1.204	0.937	1.118	1.256	0.739	1.357	0.472	1.141
bayesian regression poly d3	2.009	1.017	2.119	3.774	3.432	1.372	1.263	0.805	1.325	0.774	4.762
kernel ridge regression poly 2d	1.597	0.952	1.799	2.149	0.981	1.198	1.417	1.029	1.397	0.505	1.727
svr rbf	6.721	0.893	4.342	7.254	1.191	4.084	7.603	1.184	4.627	0.468	4.582
svr sigmoid	7.661	1.028	4.868	7.953	1.316	5.250	8.071	1.327	5.267	0.471	4.802
svr linear	16.403	7.147	3.607	4.261	2.568	1.641	6.030	13.235	1.839	2.308	1.387
svr poly 2d	80879.991	7.023E+05	263.162	746.780	2269.845	323.855	503.013	186.788	25.763	34.188	20.074
svr poly 3d	8.401E+05	6.340E+11	116.953	1539.824	1325.932	43891.161	307.171	172.504	47.519	13.749	38.943
knn uniform weights	1.886	0.744	2.819	2.209	0.925	1.936	3.425	1.147	2.941	0.424	3.964
knn distance weights	1.767	0.703	2.645	1.853	0.868	1.649	2.980	1.070	2.724	0.425	3.765
mlp regression sklearn	1.574	0.895	1.693	1.262	0.964	1.339	1.288	0.893	1.350	0.476	1.084
tensorflow neural network	7.682	4.137	5.462	8.731	11.275	5.765	8.141	9.044	5.949	6.553	5.189

Table 4.1: RMSE scores on the temperature data sets

humidity	total	indoor	outdoor	standing still	standing still indoor	standing still outdoor	walking	walking indoor	walking outdoor	office work indoor	running outdoor
no learning score	9.205	8.420	10.398	8.475	4.213	11.015	8.103	3.460	11.086	10.362	8.361
median shift	4.844	4.843	4.726	5.026	2.596	4.239	5.808	2.979	5.179	1.858	4.212
ridge regression single input	4.670	4.051	4.743	4.526	2.186	4.212	5.652	1.669	5.108	1.680	4.148
ridge regression	4.252	2.764	4.477	3.379	1.916	3.514	3.626	1.563	4.546	1.580	3.805
bayesian regression	4.251	2.764	4.475	3.385	1.945	3.502	3.633	1.589	4.513	1.551	3.943
bayesian regression poly d2	3.920	2.402	4.497	2.951	1.841	2.451	3.924	1.519	3.726	1.069	4.824
bayesian regression poly d3	5.759	2.462	7.077	7.840	5.724	2.289	3.889	1.398	3.575	2.154	24.985
kernel ridge regression poly 2d	4.288	2.506	4.323	3.584	1.955	3.697	4.114	1.523	4.320	1.223	8.152
svr rbf	7.543	3.160	6.283	8.673	2.224	6.279	9.459	2.879	6.505	2.741	6.694
svr sigmoid	8.460	4.304	7.223	9.335	2.513	7.171	10.152	2.997	7.159	3.816	7.349
svr linear	12.546	11.032	6.060	5.037	5.824	3.833	17.379	11.120	4.793	8.467	3.830
svr poly 2d	8.65E+04	6.49E+04	690.993	2486.750	2529.066	71.066	783.875	397.862	82.128	911.598	52.138
svr poly 3d	6536.084	8.44E+09	327.443	2606.576	980.015	122.192	583.011	1.22E+04	147.900	260.886	133.482
knn uniform weights	3.560	2.318	4.294	3.626	1.671	4.440	4.758	1.720	5.276	1.913	5.408
knn distance weights	3.382	2.191	4.131	3.303	1.504	3.920	4.346	1.663	4.783	1.830	4.968
mlp regression sklearn	3.618	2.189	4.287	3.272	1.813	3.378	3.604	1.715	4.316	1.349	3.565
tensorflow neural network	9.182	8.091	9.806	7.483	13.999	10.536	7.794	9.832	12.653	8.658	16.301

Table 4.2: RMSE scores on the humidity data sets

sound level	total	indoor	outdoor	standing still	standing still indoor	standing still outdoor	walking	walking indoor	walking outdoor	office work indoor	running outdoor
no learning score	6.986	7.259	6.314	8.410	11.576	4.299	4.912	6.289	2.960	5.903	9.682
median shift	7.052	7.368	6.328	6.944	8.841	3.619	4.838	6.177	2.849	7.015	5.192
ridge regression single input	6.350	7.066	3.669	5.094	5.683	2.352	4.518	5.356	2.278	5.721	4.306
ridge regression	5.164	5.770	2.362	4.173	5.101	1.977	3.486	4.567	1.551	3.429	1.293
bayesian regression	5.166	5.758	2.358	4.194	4.881	1.967	3.458	4.410	1.548	3.444	1.290
bayesian regression poly d2	4.508	7.676	2.106	3.929	5.252	1.596	3.495	4.508	1.518	2.296	1.945
bayesian regression poly d3	3.829	9.762	5.233	6.730	14.320	1.842	3.350	6.749	1.331	2.315	18.156
kernel ridge regression poly 2d	5.564	5.159	2.266	4.368	7.195	1.654	3.859	4.964	1.371	2.360	2.374
svr rbf	7.933	7.430	3.774	5.943	5.074	3.688	6.521	4.922	3.875	4.289	4.069
svr sigmoid	9.995	7.591	4.282	6.925	5.724	4.139	6.972	5.097	4.700	5.059	4.964
svr linear	16.685	51.050	3.941	8.377	13.225	2.062	13.613	12.275	1.741	10.005	2.400
svr poly 2d	4.42E+05	4.39E+05	819.632	4876.463	318.154	167.444	5.31E+04	2.07E+04	36.843	1966.976	22.143
svr poly 3d	2.26E+11	4.50E+07	224.869	1.57E+09	762.147	212.311	1.55E+04	886.347	46.630	603.874	44.641
knn uniform weights	3.369	3.247	2.465	3.312	4.061	2.189	4.314	5.006	2.228	1.052	3.297
knn distance weights	3.184	3.033	2.367	3.124	3.769	1.975	3.959	4.500	1.991	1.016	3.026
mlp regression sklearn	4.162	4.087	3.211	4.448	5.206	2.011	3.727	4.341	1.890	1.499	4.277
tensorflow neural network	8.294	7.436	12.011	11.451	17.026	12.561	11.545	15.475	13.614	6.585	18.245

Table 4.3: RMSE scores on the sound level data sets

co2 concentration	total	indoor	outdoor	standing still	standing still indoor	standing still outdoor	walking	walking indoor	walking outdoor	office work indoor	running outdoor
no learning score	799.511	1009.997	201.807	701.248	989.567	142.570	1199.141	1737.601	208.984	534.625	233.345
median shift	773.109	934.509	173.787	689.106	926.052	141.361	1100.970	1555.112	123.442	323.731	197.770
ridge regression single input	209.066	186.326	119.131	120.441	115.825	109.619	98.555	115.161	43.544	145.399	147.790
ridge regression	195.052	143.997	105.499	110.743	102.336	103.047	94.779	88.108	35.954	115.189	128.078
bayesian regression	194.749	138.404	105.834	110.253	103.146	102.847	94.902	88.137	35.993	114.868	130.237
bayesian regression poly d2	165.537	146.526	122.025	109.956	92.732	103.715	87.596	79.946	36.033	102.472	120.571
bayesian regression poly d3	226.253	188.207	163.439	128.985	124.357	101.885	72.712	67.108	33.209	117.954	326.952
kernel ridge regression poly 2d	185.390	152.170	114.799	113.064	94.005	105.216	87.696	80.227	33.986	95.800	131.671
svr rbf	169.589	168.286	118.531	115.156	120.262	105.868	94.804	120.850	46.607	135.095	152.284
svr sigmoid	167.496	174.641	120.115	117.427	124.190	110.400	96.821	124.371	46.703	139.188	153.628
svr linear	205.934	139.807	111.264	116.092	113.272	106.252	121.952	117.253	36.422	127.295	143.062
svr poly 2d	4.18E+04	2.15E+04	2031.098	3.81E+04	3.76E+04	566.513	2.31E+05	1.11E+05	766.386	2476.846	2156.487
svr poly 3d	2.24E+05	4.36E+04	5.46E+04	1.13E+08	1.97E+07	1.01E+05	6.39E+07	7.18E+10	1591.684	8.26E+04	5897.598
knn uniform weights	205.886	183.867	107.095	105.504	79.930	85.821	76.238	85.581	40.905	105.067	136.534
knn distance weights	195.644	175.237	104.500	100.580	77.893	84.481	73.049	83.480	38.010	104.233	127.335
mlp regression sklearn	185.763	141.345	107.443	108.112	95.689	101.874	80.458	90.690	34.492	98.263	141.166
tensorflow neural network	466.044	477.251	181.883	227.675	167.958	117.924	235.584	295.771	50.528	142.415	161.346

Table 4.4: RMSE scores on the CO_2 concentration data sets

tvoc concentration	total	indoor	outdoor	standing still	standing still indoor	standing still outdoor	walking	walking indoor	walking outdoor	office work indoor	running outdoor
no learning score	121.824	153.855	30.892	106.601	150.381	21.548	182.660	264.723	31.940	81.356	35.467
median shift	117.786	134.477	27.744	107.839	151.379	23.409	168.184	254.826	19.246	48.564	30.700
ridge regression single input	31.347	27.559	26.678	25.761	25.128	22.848	19.380	24.035	6.310	26.948	37.576
ridge regression	29.156	21.188	21.002	22.994	22.096	22.176	18.803	20.954	5.302	21.011	32.122
bayesian regression	29.037	20.671	21.029	23.083	21.925	22.552	18.855	20.982	5.288	20.010	32.028
bayesian regression poly d2	26.743	21.361	21.456	20.941	20.884	23.094	16.610	19.897	5.207	16.657	27.918
bayesian regression poly d3	35.576	37.387	37.090	42.059	46.897	29.298	17.514	23.631	5.085	20.213	61.198
kernel ridge regression poly 2d	28.560	22.949	21.431	20.299	19.377	20.480	16.488	20.393	4.965	17.265	28.164
svr rbf	27.882	25.547	19.830	21.929	23.019	16.749	15.692	26.391	6.927	23.628	25.566
svr sigmoid	25.012	23.887	20.036	19.821	24.865	16.968	15.234	26.091	6.883	21.017	24.555
svr linear	41.193	59.068	16.716	29.024	34.758	16.710	31.029	48.020	5.623	22.436	25.069
svr poly 2d	3.80E+04	7.72E+04	843.350	9887.347	1999.904	348.885	2.70E+04	3857.592	245.475	2354.231	256.314
svr poly 3d	3.68E+04	1.03E+06	2103.190	1.00E+08	2081.681	884.275	1378.969	3.85E+04	91.789	3082.637	1104.866
knn uniform weights	28.906	24.860	25.343	23.947	21.745	22.504	13.890	17.963	6.050	19.149	37.141
knn distance weights	27.512	23.705	23.969	22.228	18.280	21.477	13.570	17.006	5.760	18.695	34.561
mlp regression sklearn	27.878	20.770	22.159	21.424	20.289	26.248	14.352	17.311	5.528	15.691	31.609
tensorflow neural network	31.998	25.930	22.668	25.750	27.832	20.449	20.658	31.980	6.317	22.900	29.970

Table 4.5: RMSE scores on the TVOC concentration data sets

overall scores	mean	mean no aggregation	median	median no aggregation
no learning score	166.349	151.441	9.682	10.689
median shift	148.244	128.606	6.328	5.684
ridge regression single input	32.633	29.066	5.356	5.232
ridge regression	28.131	24.579	4.477	4.176
bayesian regression	28.038	24.623	4.410	4.176
bayesian regression poly d2	**26.766**	22.915	4.497	**4.117**
bayesian regression poly d3	37.644	35.170	6.749	6.237
kernel ridge regression poly 2d	27.454	23.306	4.368	4.642
svr rbf	31.698	29.087	7.254	5.677
svr sigmoid	32.423	29.933	7.661	6.304
svr linear	35.527	29.957	13.225	10.562
svr poly 2d	4.66E+04	6457.091	1999.904	373.374
svr poly 3d	1.71E+10	2.39E+09	2081.681	885.311
knn uniform weights	28.179	23.600	4.314	4.723
knn distance weights	26.902	**22.555**	**3.959**	4.210
mlp regression sklearn	27.152	24.011	4.277	4.297
tensorflow neural network	56.679	42.488	12.653	14.737

Table 4.6: Mean and median of the RMSE scores, no aggregation excludes data sets that combine data, i.e. indoor or walking

4.1 Performance of the Ground Truth Estimation Algorithms

The scores of the different learning algorithms can be found in six tables. There is a table with scores for all temperature data sets (table 4.1), a table for all humidity data sets (table 4.2), a table for all sound level data sets (table 4.3), a table for all CO_2 concentration data sets (table 4.4), a table for all TVOC concentration data sets (table 4.5) and a table with the median and median scores (table 4.6).

Baseline Algorithm Our baseline algorithm, which shifts the data by the median relative error, shows already a good performance considering the temperature (table 4.1) and humidity (table 4.2) data sets. On these data sets some of the RMSE scores where reduced by up to 50%. On the CO_2 (table 4.4) and TVOC concentration (table 4.5) data sets nearly no improvement can be seen. The scores on the sound level data set (table 4.3) is even a bit worse compared to the base score.

Ridge Regression and Bayesian Regression The two Ridge Regression and the Bayesian Regression algorithms performed very well considering the over all scores (table 4.6). On the temperature data set (table 4.1) they are under the top performing algorithms. On the other sensor type data sets they still perform very well, especially on the different walking and running contexts.
The Ridge Regression algorithm with one-dimensional input performs the worst of the three algorithms, which indicates that other sensor values can give hints on the error of an other sensor value. The other two algorithms perform very similar. This is no surprise as the difference between the two is only that in Ridge Regression the gamma parameter is chosen via cross validation and in Bayesian Regression the gamma parameter is learned within its mathematical model.
Worth mentioning is the fact that Bayesian Regression is the fastest algorithm to train.

Bayesian Regression with Data transformed into Polynomial Space of Degree 2 and 3 On the temperature data sets (table 4.1) the scores are comparable to the scores of the normal Bayesian Regression, which means both variants are under the top performing algorithms. On the humidity data set (table 4.2) the two algorithms are under the top performing algorithms as well. On the data sets of the other sensor types the two algorithms are always outperformed by other algorithms, but they are not far off on the data sets of the walking and running contexts.
The Bayesian Regression with data transformed into polynomial space of degree two is one of the best performing algorithms considering the median and mean of all scores (table 4.6).

Kernel Ridge Regression with Polynomial Kernel of Degree 2 In most cases the performance of this algorithm is close but not equal to the top performing algorithms. In some rare occurrences on the CO_2 (table 4.4) and TVOC concentration data sets (table 4.5) the algorithm has the best score, for example on the TVOC concentration walking outdoor data set.

SVR with RBF, Sigmoid and Linear Kernel In nearly all cases the performance of the three algorithms is not good. There are cases where the score is even worse than the score of the data without learning. On the other hand, on the TVOC concentration data sets (table 4.5) the algorithms performs better, giving the best results of all algorithms for standing still outdoor and running outdoor.

SVR with Polynomial Kernel of Degree 2 and 3 These two algorithms have problems running on the data sets. The problems are so severe that the number of iterations the algorithms can perform has to be limited to 10'000 to even get a result in a reasonable time. Due to this limitation the two algorithms show by far the worst performance of all algorithms.

KNN with Uniform and Distance Weights The two algorithms show good performance on aggregated data sets (total, indoor, outdoor, standing still and walking). Standing still indoor and office work are also strong points of the two algorithms. On the CO_2 and TVOC concentration data sets they perform better than most of the other algorithms.
Considering the mean and the median of all scores the KNN variant which uses distance weights is under the top two performing algorithms.

Multi-Layer Perceptron Neural Network (scikit-learn implementation) Looking at the over all scores (mean and median of all scores) (table 4.6) the performance of this algorithm is good and is always close to the best performing algorithms in the individual cases. On the CO_2 (table 4.4) and TVOC concentration data sets (table 4.5) the performance is not as good as on the data sets of the other sensor types. On the aggregated data sets (total, indoor, outdoor, standing still and walking) the performance is good as well.
A big part of the good performance comes from the lbfgs optimizer. Changing to an other optimizer leads to significant worse scores.

Multi-Layer Perceptron Neural Network (TensorFlow) This algorithm does not perform good. It seems that the algorithm suffers from some oscillation and is on the edge to diverge (changes to the configuration leads to divergence). Despite some testing with different optimizers the TensorFlow framework provides, we can not find another optimizer which performs better than the momentum optimizer. The lbfgs optimizer, used by the neural network implemented with scikit-learn, is not available in the TensorFlow framework.

4.2 Performance of the Truth Finding Algorithm GTM

Where the complete GTM output of a run with up to 10 and a run with up to 20 claims and different source configurations can be found is written in the appendix A. A few example outputs can be found in listing 4.1, which serves as an example for the problems GTM has on our data set.

4.2.1 Performance Analysis of GTM

The performance of GTM is disappointing. Besides some rare cases when dealing with TVOC concentration data, GTM is not able to challenge the top performing algorithms from the ground truth estimation (section 3.2.1 and 4.1). The mathematical model behind GTM can give a possible explanation for this performance. The real truths we provide GTM, are only used as a first estimation of the Gaussian distribution. The problem is, the more claims the different sources make the more likely it is for the model that the claims are the truth (or at least closer to the truth). Strictly speaking, the Gaussian distribution deviates from the real truth with every iteration of the algorithm if the sources do not provide claims that are telling the truth or do not provide claims that are close to the truth. And indeed, examples where more claims lead to worse results can be found (listing 4.1). The assigned quality levels are also worse in these cases (lower is better).

```
input explanation:

number of each sources has the form [A, B, C]
A always corresponds to the number of standing still sources,
B always corresponds to the number of walking sources and
C always corresponds to the number of running or office work sources
(depending if indoor or outdoor)

number of claims found M/N means M claims found, was looking
for up to N claims

the different sources get a quality level assigned, smaller is better

##########################################

indoor results

        co2 level

                number of each source: [2, 2, 2]
                number of claims found: 10/10
                RMSE of found truths:                   346.920797031

                        standing_still_indoor_0:        6.61178228027
                        standing_still_indoor_1:        7.41559129189
                        walking_indoor_0:               6.33696723317
                        walking_indoor_1:               6.33696723317
                        office_work_indoor_0:           6.33696723317
                        office_work_indoor_1:           6.33696723317

                number of each source: [2, 2, 2]
                number of claims found: 15/20
                RMSE of found truths:                   368.300282622

                        standing_still_indoor_0:        8.9250750701
                        standing_still_indoor_1:        9.22567149362
                        walking_indoor_0:               8.2326023983
                        walking_indoor_1:               8.2326023983
                        office_work_indoor_0:           8.2326023983
                        office_work_indoor_1:           8.2326023983


outdoor results

        temperature
```

```
number of each source: [3, 3, 3]
number of claims found: 10/10
RMSE of found truths:                    2.55184711683

        standing_still_outdoor_0:        7.27497877048
        standing_still_outdoor_1:        6.77568064148
        standing_still_outdoor_2:        6.65503307511
        walking_outdoor_0:               6.39287710099
        walking_outdoor_1:               6.39287710099
        walking_outdoor_2:               6.39287710099
        running_outdoor_0:               6.39287710099
        running_outdoor_1:               6.39287710099
        running_outdoor_2:               6.39287710099

number of each source: [3, 3, 3]
number of claims found: 18/20
RMSE of found truths:                    2.75773715353

        standing_still_outdoor_0:        10.247715894
        standing_still_outdoor_1:        9.83468592154
        standing_still_outdoor_2:        9.75953450452
        walking_outdoor_0:               9.20137203457
        walking_outdoor_1:               9.20137203457
        walking_outdoor_2:               9.20137203457
        running_outdoor_0:               9.20137203457
        running_outdoor_1:               9.20137203457
        running_outdoor_2:               9.20137203457
```

Listing 4.1: Examples where GTM performs worse with more claims than with less claims, assigned quality levels are also worse (lower is better) with more claims

4.2.2 Analysis of the Context Handling by GTM

As GTM assigns quality levels to the sources (section 3.3.1) and we create the sources from the different contexts (section 3.3.2), GTM should give us some insights how the different context have an influence on the algorithm. Unfortunately, the assigned quality levels are very close to each other (listing 4.1) which does not allow us to say a certain source/context is favored. In other words, it most likely makes no difference from which context the source has its data from and GTM can most likely not handle the different contexts. This directly contradicts our observation (section 2.3.2) where we can observe that the contexts have an influence on the data quality.

4.3 Conclusion

In this chapter we have seen that ground truth estimation works better if not only one sensor type is considered. The Ridge Regression algorithm with all five sensor types outperforms itself if trained with only one sensor type.
For the temperature and humidity data sets our baseline algorithms which shifts the data by the median of the relative errors already shows good results. With a bit more work in training by using Bayesian Regression and transforming the data into polynomial space of degree two most of the errors introduced by the contexts can be removed.
The K nearest neighbors algorithms using distance weights is able to handle combined data sets (i.e. both walking indoor and walking outdoor combined). Also the CO_2 and TVOC concentration data sets are handled well by this algorithms. Together with Bayesian Regression with transformed data into polynomial space of degree two KNN is the best performing algorithm on our data sets.
The truth finding algorithm GTM is not able to recognize the the relationship between context

and data quality. To assume that the claims are represented by a Gaussian distribution with the ground truth as mean seems to be wrong.

Chapter 5

Conclusion

In this chapter we draw our conclusion of the thesis. In section 5.1 we briefly summarize the main findings of the thesis. Section 5.2 lists points which can be improved in a future work. Lastly, section 5.3 lists the take-away messages how to handle data coming from a mobile crowdsensing context.

5.1 Summary

In this thesis we have conducted our own mobile crowdsensing measurements in a variety of different contexts. These contexts were standing still, walking, running and office work. The person conducting a measurement acted according to the context the measurement should have had attached to it. During a measurement the temperature, the humidity, the sound level, the CO_2 concentration and the TVOC concentration were logged.
In the investigation of the collected data we observed the existence of a relationship between context and data quality. With this relationship observed, we tested different learning algorithms to estimate the ground truth and correct the errors introduced by the context. For comparison we tested GTM, an algorithm from the truth finding algorithm family. Truth finding algorithms are developed with the idea to discover the truth in the data provided by multiple sources. We observed that there exist algorithms suitable for the task of estimating the ground truth. Bayesian Regression with the input data transformed into a polynomial space of degree two and the K nearest neighbors algorithm are two algorithms which are able to correct most of the errors in our data sets. The truth finding algorithm, GTM, is not able to deal with the different contexts and can not recognize the relationship between context and data quality. All contexts get similar quality levels assigned by GTM, which contradicts our observation to the relationship between context and data quality.

5.2 Future Work

A future work can improve this thesis by considering the season in which the measurements will be performed. We performed the measurements from mid to end of November. The season could also have had an effect on the data we collected. A feature work can collect data over all seasons to eliminate this variable from the data.
It is possible that, in the future, affordable air quality sensors will get a significant improvement of their quality. In case this happens it makes sense to re-do the measurements of the two air quality metrics as the air quality sensor we used most likely suffered from its quality.

5.3 Take-Away Messages

In the following paragraphs we list take-away messages for someone handling data coming from a mobile crowdsensing context.

Context Matters The first key take-away is that context has an influence on data quality as we have seen in section 2.3.2. There is more to the context as just the activity of a participant, the environment (i.e. outdoor or indoor) plays also an important role. We suggest to keep in mind that there exists this relationship if starting a project with such data.

Tackle the Influence of Context The second key take-away is that there are simple learning algorithms which work quite well for estimating the ground truth of context affected data (section 4.1). We suggest starting with having a look at the description of the algorithms we have used in section 3.2.1 and then testing some of them on the data set.

Testing Truth Finding Algorithms and Ground Truth Our third take-away is to carefully test a truth finding algorithm before using it. It is possible, that a truth finding algorithm is not able to perform well on a particular data set while on other data sets the performance is as desired. GTM, the truth finding algorithm we tested is, for example, not able to handle the different contexts in our data (section 4.2.2).
It is also important to use meaningful ground truths for testing algorithms. Assuming that the ground truths follow a certain mathematical model can lead to a wrong test result, especially if the tested algorithm, by chance, uses the same mathematical model.

Appendix A

Data and Source Code

On request, the data from the measurements and the source code of the software used in this thesis is available.

A.1 Data

The data from the overnight measurements can be found in:

`\Data\Measurements\Overnight_Measurement`

The aggregated data from the measurements in context as well as the statistics of this data can be found in:

`\Data\Measurements\Measurements_in_ Context`

The GTM output data from two runs (one up to 10 claims and one with up to 20 claims per truth) can be found in:

`\Data\Ground_Truth_Estimation_Results\GTM_Results`

The performance scores of the ground truth estimation algorithms can be found in:

`\Data\Ground_Truth_Estimation_Results\Algorithms_Performance`

A.2 Source Code

The source code for the modified Android app (mostly DemoMeasurementActivity.java) can be found in:

`\Source_Code\thunderboard-android-app`

The source code for the modified Thunderboard firmware (radio_ble.c) can be found in:

`\Source_Code\thunderboard-firmware\soc-thunderboard-sense`

The source code for the script reading the sensor values from a Thunderboard connected to a computer can be found in:

`\Source_Code\serial-port-reader`

The source code for the preprocessing of the measurement data can be found in:

`\Source_Code\measurement_analysis`

The source code of the different learning algorithms and the truth finding algorithm, GTM, can be found in:

`\Source_Code\truth_estimation`

Appendix B

Bibliography

[1] Context-Aware Data Quality Estimation in Mobile Crowdsensing,
Shengzhong Liu, Zhenzhe Zheng, Fan Wu, Shaojie Tang and Guihai Chen,
IEEE Conference on Computer Communications INFOCOM 2017,
May 2017, Atlanta, GA, USA

[2] A Survey on Truth Discovery,
Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan and Jiawei Han,
ACM SIGKDD Explorations Newsletter Volume 17 Issue 2,
December 2015, New York, NY, USA

[3] Participatory Sensing or Participatory Nonsense?: Mitigating the Effect of Human Error on Data Quality in Citizen Science,
Matthias Budde, Andrea Schankin, Julien Hoffmann, Marcel Danz, Till Riedel and Michael Beigl,
Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies Volume 1 Issue 3,
September 2017, New York, NY, USA

[4] Complex Human Activity Recognition Using Smartphone and Wrist-Worn Motion Sensors,
Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten and Paul J. M. Havinga,
PMC Us National Library of Medicine, National Institutes of Health, Sensors Journal 2016 Volume 16 Issue 4,
March 2016, Basel, Switzerland

[5] Apple Watch Series 3 Specifications,
Apple, Website,
https://support.apple.com/kb/SP766?viewlocale=en_US&locale=de_CH,
Last visited on the 11th of February 2018

[6] Garmin vivoactive 3 Specifications,
Garmin, Website,
https://buy.garmin.com/en-US/US/p/571520#specs,
Last visited on the 11th of February 2018

[7] NoiseTube Crowdsensing Project,
NoiseTube, Website,
http://www.noisetube.net/index.html#&panel1-1,
Last visited on the 11th of February 2018

[8] OpenSense 2 Crowdsensing Project EPFL Lausannne,
EPFL Lausanne, Website,
http://opensense.epfl.ch/wiki/index.php/OpenSense_2,
Last visited on the 11th of February 2018

[9] Simplicity Studio 4,
Silicon Labs,
https://www.silabs.com/products/development-tools/software/simplicity-studio,
Last visited on the 29th of January 2018

[10] Source code of the official Thunderboard Android app,
Silicon Labs, GitHub repository,
https://github.com/SiliconLabs/thunderboard-android,
Last visited on the 29th of January 2018

[11] Ridge Regression,
scikit-learn.org, documentation,
http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html,
Last visited on the 8th of February 2018

[12] Bayesian Regression,
scikit-learn.org, documentation,
http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.BayesianRidge.html,
Last visited on the 8th of February 2018

[13] Poynomial Features,
scikit-learn.org, documentation,
http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html,
Last visited on the 8th of February 2018

[14] Kernel Ridge Regression,
scikit-learn.org, documentation,
http://scikit-learn.org/stable/modules/generated/sklearn.kernel_ridge.KernelRidge.html,
Last visited on the 8th of February 2018

[15] Support Vector Regression,
scikit-learn.org, documentation,
http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html,
Last visited on the 8th of February 2018

[16] K Neighbors Regression,
scikit-learn.org, documentation,
http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html,
Last visited on the 8th of February 2018

[17] Multi-Layer Perceptron Regressor,
scikit-learn.org, documentation,
http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html,
Last visited on the 8th of February 2018

[18] TensorFlow,
TensorFlow website,
https://www.tensorflow.org/,
Last visited on the 8th of February 2018

[19] Root Mean Squared Error (RMSE) Explanation,
Professor Susan Holmes,
Introductory Statistics,
Stanford University, CA, USA,
http://statweb.stanford.edu/ susan/courses/s60/split/node60.html,
Last visited on the 19th of February 2018

[20] A Probabilistic Model for Estimating Real-valued Truth from Conflicting Sources,
Bo Zhao and Jiawei Han,
Proceedings Proceedings of the VLDB workshop on Quality in Databases QDB 2012