Real Time Thermal Comfort Prediction

J. Dinal Herath, Zhihua Li
Department of Computer Science, SUNY Binghamton, USA
jherath1@binghamton.edu, zli191@binghamton.edu

1 INTRODUCTION

Wireless sensor network has wide application prospects in smart home system. Among which Heating, Ventilation and Air Conditioning (HVAC) systems that have the capability to automatically adjust the thermal comfort level in a given indoor setting has great real world potential. Current systems combine real time sensing with actuation to create these systems. In our project, we aim to push this boundary by investigating the possibility of stacking real time prediction on top of an existing HVAC system, thereby improving overall performance. To this end, our project is carried out in two phases. In phase one we create a temperature acquisition system using wireless sensor network based on TinyOS. Here wireless node module and sensor module are used and TinyOS operating mechanism to build a multi-hop network that we use to collect temperature datasets. In the second phase, we experiment with potential predictive models spanning both deep learning and machine learning approaches and finally identify that linear regression, the machine learning model has the best fit in terms of both predictive accuracy and computational overheard in this scenario.

2 IMPLEMENTATION

In this section, we present the implementation details pertaining to building a sensor network for temperature data collection, details regarding the datasets used and preprocessing steps and our experimentation with different predictive models.

2.1 Data Collection

Due to the limit transmission distance of single TelosB mote, we build a prototype of indoor and outdoor sensing network with 3 TelosB nodes: Sender 1 is used to sense environment temperature and humidity and send the data packages to Sender 2, Sender 2 receives them and relay to Basestation. The Basestation takes all the packages and send them to computer via serial, then the computer can display and save the data. Under this sensing network, we can collect temperature and humidity of different locations and the Base station can successfully communicate with the Senders from a longer distance. Which can be wildly used in real sensing situations. Figure 1 shows the data collection structure.

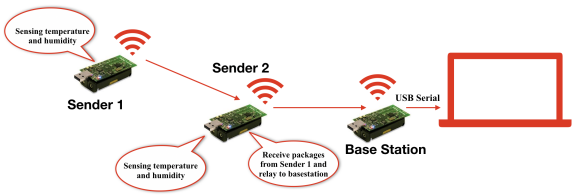


Figure 1: Multi-hop Data Collection Network

All the TemosB motes are based on TinyOS, and the TelosB motes are integrated with Sensirion SHT11 module used for temperature and humidity sensing. In TinyOS, the Read interface is used for data reading. As for the wireless communication, we need interfaces like SplitControl, Packet, AMsend. SplitControl is responsible for starting the wireless communication module, Packet is used for data package processing, AMSend is for data transmission, Timer is used to trigger interrupts therefore we can read the data in a certain frequency. In order to avoid data conflicts, we need to determine the mote address before compiling, and use AMSend to send the data to a certain address instead of broadcasting all the masseges. Figure 2 shows the control flow of the relay node.

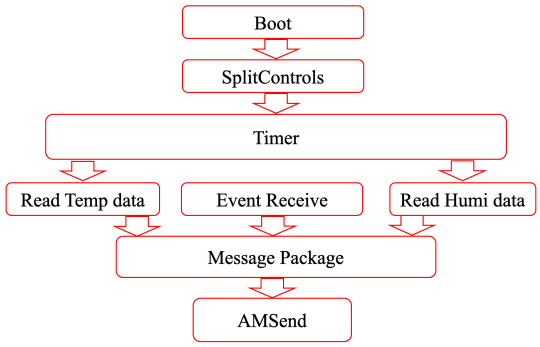


Figure 2: Control Flow of Relay Node

As for the Mote-PC serial communication, BaseStation acts as a bridge between the serial port and radio network. When it receives a packet from the serial port, it transmits it on the radio; when it receives a packets over the radio, it transmits it to the serial port. Because TinyOS has a toolchain for generating and sending packets to a mote over a serial port, using a BaseStation allows PC tools to communicate directly with mote networks. For the upper computer side, we use Java-based infrastructure for communicating with motes, read the package into a text file and print them in real time. Figure 3 shows the upper computer system display.

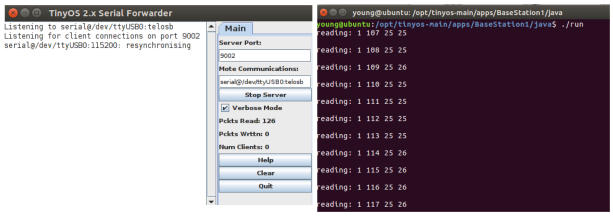
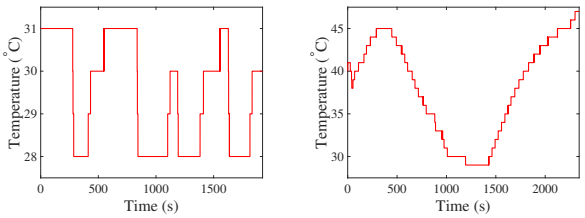


Figure 3: Upper Computer System Screen Shot

2.2 Datasets and Data preprocessing

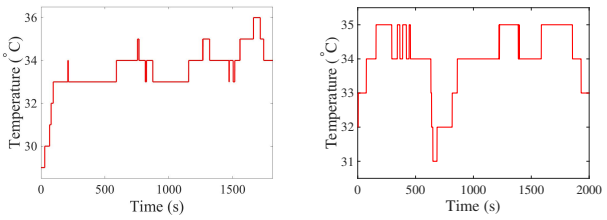
For the purpose of our experimentation, we collect indoor temperature samples across two categories, namely based on location and sampling rate. Initially, we recorded temperature variations within the computer science department building complex and within an apartment setting. Since our objective is to investigate a predictive model that works with little strain on an existing HVAC system, all datasets shown here are collected while some HVAC system or heating system is functional. Figure 4 shows the datasets collected in this fashion, with Figures 4(a) and 4(b) depicting temperature variations in the computer science department and apartment setting respectively. Each sample shows temperature variations with a time-step of 250 msec for a total durations of approximately 30 minutes.



(a) CS Building (b) Apartment

Figure 4: Datasets based on Location

Additionally, as shown in Figure 5 we collected datasets based on different sampling rates. Figures 5(a) and 5(b) shows temperature variations in the computer science building for sampling rates of 100 msec and 1 sec respectively. Overall, the dataset contains temperature samples for time duration of approximately 30 minutes. We note that there is limited erratic behavior in temperature variations shown in these samples, and attribute it to the functionality of the heating system or HVAC in play.



(a) 100 msec (b) 1 sec

Figure 5: Datasets based on Sampling Rate

2.3 Predictive Models

In this section, we present the predictive models we designed for the purpose of predicting future temperature variations. Identifying the most suitable predictive model for our purpose proved to be the biggest challenge in the entire project. Therefore, we approached the temperature prediction problem utilizing two types of predictive models–i) a machine learning approach and ii) a deep learning approach respectively.

For our machine learning approach we build a predictive model based on linear regression. Linear Regression is an online model that is capable of dynamically predicting temperature variations on the go, however with the possible limitation of not being able to infer complex variations in the time series. In contrast, we employ an offline deep learning model based on Recurrent Neural Networks (RNNs) which are capable of inferring complex variations, however these models require extensive computational resource thus might incur additional stress to an existing HVAC system when deployed. Since we did not know how temperature would vary in an indoor setting given some thermal management system, we experimented with these two contrasting approaches as predictive models.

2.3.1 Machine Learning Approach. For our machine learning approach, we utilize a model named linear regression which is about mapping a series of inputs x_T to a series of outputs y_T with a predefined linear relationship as shown in equation 1 below. Since the mapping is linear, the relationship between x_T and y_T can be given using the coefficients β_0 and β_1. With reference to this problem, a sliding window of one time-step is used to obtain a sequence of inputs of length 50 for x_T which is used to obtain predictions for 20 time-steps into the future. With this model we make an implicit assumption that near future variations in temperature will follow a linear fashion.

$$y_T = \beta_0 + \beta_1 x_T \tag{1}$$

The purpose of the learning algorithm is to determine the best fit for β_0 and β_1 that maps x_T to y_T. Though simplistic in nature, this model is extremely fast in convergence compared to deep learning and requires less computational resources as well. The gradient descent algorithm is used to optimize the coefficient mapping between inputs and outputs, we refer the reader to Raschka et al [2] for more information regrading the inner workings of the algorithm used.

2.3.2 Deep Learning Approach. For our deep learning approach, we utilized an architecture named Recurrent Neural Networks (RNNs) which are a type of deep neural network with a sequence based processing specialization, which in our case turns out to be a sequence of temperature variations. RNNs achieve this by having internal memory which remembers state through parameter sharing. The architecture of the RNNs shown in figure 6, which functions as the building block of our proposed model. Let $X_t = [x_{t-2}, x_{t-1}, x_t]$ be the input vector and $Y_t = [y_{t-2}, y_{t-1}, y_t]$ be the corresponding output vector to the illustrated model.

As per figure 6 the unfolded structure of the RNN shows the calculation done at each time step t. The hidden state h_t which serves as memory is calculated using the hidden state maintained h_{t-1} and the input x_t using the following equation:

$$h_t = \sigma_h \left(W_{xh} x_t + W_{hh} h_{t-1} + b_h \right) \tag{2}$$

Where W_{xh} denotes the weight matrix from the input layer to the hidden layer, W_{hh} the weight matrix between the hidden states of two consecutive time steps t and $t-1$, b_h the bias vector of the hidden

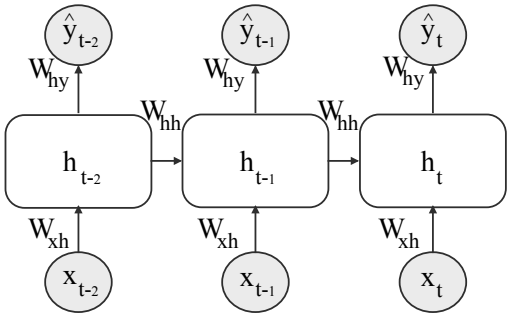


Figure 6: RNN architecture

layer and σ_h the activation function for the hidden state. Using the hidden state at instance t the network output can be obtained by:

$$y_t = \sigma_y \left(W_{hy} h_t + b_y \right) \tag{3}$$

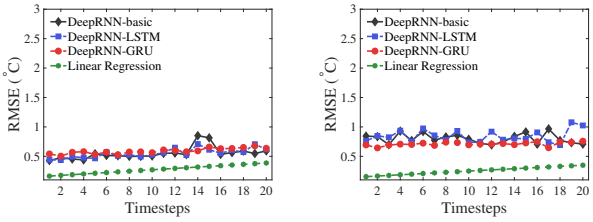
Where W_{hy} is the weight matrix from the hidden layer to the output layer, b_y its corresponding bias vector and σ_y the activation function. The parameters of the above equations are trained iteratively using the back propagation algorithm. While RNNs are versatile in sequence prediction, they suffer from two major drawbacks–namely vanishing gradient and the exploding gradient. This hinders its ability to learn long-term dependencies. To handle this problem, long short-term memory (LSTM) and gated recurrent unit (GRU) architectures have been proposed which are RNNs that create paths through time with derivatives that doesn't vanish or explode. We refer the reader to Benjio et al [1] for more in depth information about LSTMs and GRUs. Given all these possibilities, we test three types of deep learning models, namely i) a basic RNN model, ii) RNN-LSTM and iii) RNN-GRU as further shown in sections below.

3 EVALUATION

In this section, we present the evaluation of the predictive models used. For the purpose of evaluating the predictive performance of the models described in the section above, we use the root-mean squared error (RMSE) given in equation 4 below.

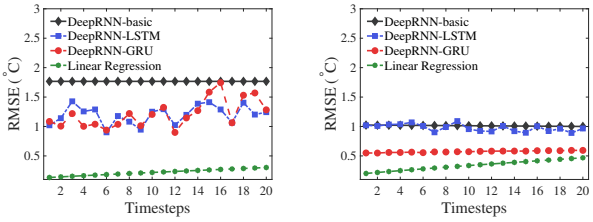
$$RMSE_j = \sqrt{\frac{\sum_{i=1}^{h} \left(\hat{y}_{ij} - y_{ij} \right)^2}{h}} \tag{4}$$

Here, Let y_{ij} be the i^{th} test sample for the j^{th} prediction step, and $\hat{y}_{ij}$ be the predicted value of y_{ij} and h the number of test samples. Overall, from our evaluation we note that the linear regression model is the best fit for our purpose of predicting temperature variations in real time. Figures 7-8 show the RMSE results for all datasets considering both the deep learning models and the machine learning approach. From the results here we note that the machine learning solution consistently outperforms the deep learning models. We note that the overall lower complexity in the datasets to be a possible reason behind this performance variation. A more important factor to note is that while the performance of all models show a predictive error < 1 in most cases, linear regression provides predictions at a speed of 0.2 msec per prediction where as the deep learning models take approximately 3-4 hours for training. Therefore, for the purpose of this project we determine linear regression to be a suitable model with high accuracy and low overhead to be stacked upon an existing HVAC system.



(a) Computer Science Building (b) Apartment

Figure 7: RMSE results based on location



(a) 100 msec (b) 1 sec

Figure 8: RMSE results based on sampling rate

4 CONCLUSION

In this project we aimed to identify a predictive model that can be utilized in tandem with current HVAC systems in order to incorporate prediction as part of HVAC functionality. To this end, we created a sensor network system based on TelosB motes and collected our own datasets across different dimensions. Afterwards, we tested the applicability of both deep learning and machine learning for this prediction task and conclude that the lightweight linear regression model is the best fit for our situation based on predictive error and potential incurred overhead in computation and thus is presented as our final contribution. (Figure 9 shows the final predictions vs actual temperature variations for the machine learning model in green)

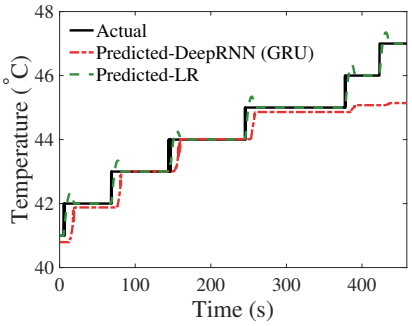


Figure 9: Actual vs Predicted

REFERENCES

[1] Yoshua Bengio, Ian J Goodfellow, and Aaron Courville. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[2] Sebastian Raschka and Vahid Mirjalili. 2017. *Python machine learning*. Packt Publishing Ltd.