Indoor Location Prediction through Modeling of Human Spatiotemporal Behavior

Submitted in partial fulfillment of the requirements for the degree of

Doctor in Philosophy

in

Electrical and Computer Engineering

Christian Koehler

M.S., Computer Science, RWTH Aachen University

Carnegie Mellon University

Pittsburgh, PA

August, 2015

Copyright

Acknowledgments

This dissertation would not have been possible with the help of colleagues, friends, and family who supported me, endured the ups and downs of the process with me, or offered me words of critique and encouragement. Somebody once told me a PhD is not a race, but a marathon and you will stumble along the way. I was fortunate enough to have many amazing people in my life who pushed me to see the end of my journey.

First and foremost I want to thank Anind Dey who, so many years ago, took a chance with a random foreign student and allowed me to work on my own projects and experience so much. Over the years you were my mentor, friend, and confidant. Your encouragement and critique pushed me to always try to do my best so my work and my career move forward. This work is very much a collaboration that would not have been possible without you.

Another big thanks is due to Jennifer Mankoff and Ian Oakley who guided me with their insightful words and occasional prodding through the process of getting a PhD. You picked me up when I stumbled and pushed me forward to the end. Your friendship and counsel over the last six years allowed me to reach the finish line of my journey.

I would also like to thank the rest of my committee, Dan Siewiorek and Oliver Brdiczka for their valuable feedback and support. Your insights helped me to shape my work.

One person I would especially like to thank is Queenie Kravitz. You were always there to listen to me be it times I was exited about good things or down because of something bad. Your wise counsel always gave me perspective and you allowed me to reflect on myself without judgment. I always liked to say that you are my unofficial counselor who helped me to navigate the more emotionally stressful situations in my PhD career.

Many people allowed me to talk through ideas, but I would especially like to thank Afsaneh Doryab and Nikola Banovic for their continued support. You allowed me to work through algorithmic problems and put me on the right path for a elegant solution. I would also like to thank Christine Bauer for allowing me to interrupt every time I'm having a dissertation writing question.

My journey towards a PhD was also enriched by a number of friends who made my time more enjoyable and allowed me to relax for a while. I would like to thank: Pascal Meyer, Nitesh Goyal, Dominik Lenhard, Leanne Davis, Iryna Zhurak, Michael Dagitses, Adrian Reyes, Kevin Huang, Rebecca Gulatta, Alyson Youngblood, Tatiana Vlahovic, Dan Tasse, Afsaneh Doryab, Nikola Banovic, Chloe Fan, Denzil Ferreira, Eija Haapalainen, Michelle Scott, and Lucas Pereira.

A special thanks also goes to Dan Siewiorek, Asim Smailagic, and Mahadev Satyanarayanan for the opportunity to teach with you. Your example showed me how to motivate and lead students and also taught me the virtue of patience.

I would also like to thank Samantha Goldstein, the former Electrical and Computer Engineering Department's PhD program coordinator, and Susan Farrington, the former Electrical and Computer Engineering Department's career consultant. Your continued support over the years allowed me to navigate the difficult PhD process.

Many thanks also goes to Jeffrey Pierce and David Nguyen for giving me a better insight into what it means to be a PhD-level researcher in industry. Your example motivated me to search for opportunities in the private sector.

One person who made it possible for me to start my PhD was Walter Unger. Thanks so much for recommending me to the Carnegie Mellon University's PhD program so many years ago.

A lot of this work would not have been possible without the support and help from Carnegie Mellon University's facility management team, so a special thanks also goes to Marty Altschul and Jim Skees.

Lastly I would also thank my family for their loving support over the years. Without you this dissertation would not have been possible.

I would also like to thank the National Science Foundation and Portuguese Foundation for Science and Technology for the financial support I received in support of my studies under grants IIS- 1227495, IIS-1217929, and SFRH/BD/70428/2010.

Abstract

The ability to capture people's location within large indoor spaces (such as office buildings, university campus buildings, or hospitals) and to use this information to predict when and where they will go next is a potentially powerful computational tool. Indoor location prediction algorithms could enable smarter automated receptionists, support ad-hoc meetings whilst occupants are transitioning between locations, or improve room temperature control. However, predicting occupant indoor locations and when occupants will transition between these locations is challenging. Existing systems that monolithically approach this task perform poorly, which is likely because occupants' spatiotemporal routines within the interior of a building, such as a workspace, can be very complex.

To address this problem, this work explicitly models the behavior of occupants using a range of spatiotemporal features such as time of day, previous locations, or day of the month. We developed a pattern extraction algorithm, ABC-Pattern-Extract, based on Conditional Frequent-Pattern Trees (FP-Trees) that describes a person's behavior as a collection of spatiotemporal FP-Trees varying depth. Using 53 occupancy traces from a four-month data collection our algorithm is able to extract on average 1011.6 FP-Trees with an average depth of 5.3. Due to computational restrictions put in place on the algorithm, ABC-Pattern-Extract was able to create these trees in on average less than 25 minutes. To evaluate our approach we show how an extracted set of patterns can be used to accurately predict the future location of an occupant through a secondary predictive algorithm, ABC-Pattern-Predict. To show the impact of the algorithm in a real world scenario we chose efficient room temperature control and simulated the energy impact of ABC-Pattern-Predict. Our evaluation shows that ABC-Pattern-Predict achieves a very high accuracy of 92.56% and an increase in Recall by 10.93% over the current state of the art, PreHeat. The high accuracy and recall improvement over PreHeat leads to an estimated energy consumption reduction of 16.1%, while maintaining the same level of comfort.

Table of Contents

List of Tables

List of Figures and Illustrations

1. Introduction

The ability to capture people's location within large and complex indoor spaces (such as office buildings, university campus buildings, or hospitals) and to use this information to predict when and where they will go next is a potentially powerful computational tool. Indoor location prediction algorithms can improve services, making them more adaptable to human needs, or even provide services that were not possible earlier. To highlight this lets consider the scenario of Bob, a mid-level executive in a large technology company. As a father of two young school kids, Bob has a fairly stable work routine: he usually arrives in the office at 7:00AM after dropping his kids off at school and stays till 5PM, in time to get back home for dinner with his family.

On one of his usual workdays in the middle of January, Bob drives in to the office and realizes he has to meet with Alice to talk about the latest developments in a project they both work on. Alice works in a different building on the company's campus, so he is unsure when she will be available today and he emails her to setup a meeting. Upon arriving in his office he notices that the temperature control system did not heat his space since he is one of the few people in the office that arrive this early. Even though his office has a room-level temperature control, the static control schedules that facility management has set for the building assume that the majority of the employees arrive around 8AM. Bob increases the set temperature by 8° F and leaves his jacket on until his office is heated up. After working till 10AM he leaves his office for several meetings and lunch with some clients and returns to his office at 2PM. He is glad that at least now the temperature is at the right setpoint, but he wonders how much the temperature control impacts the environment since it had to keep his office heated for four hours even though he was out of his office, attending meetings he has on a regular basis. He makes a note to bring this up with the green building manager his company hired.

Not having heard back from Alice he decides to drop by her office since a project deadline is coming up and he needs to make sure everyone is on the same page. Unfortunately she is not in her office and nobody can tell him when she is due back. Frustrated, he goes back to his office and makes a mental note to set up a meeting earlier the next time. After some productive hours he wonders why the package he expected did not arrive yet, so he calls the company's mail desk. Trying to stay in touch with the newest developments the company recently introduced MailBot, a robotic mail delivery system that promised to be more reliable and faster than the old mail guy. Unfortunately for many of his colleagues the opposite is true. The mail desk lets him know that the robot was at his office, but he was not there so it could not deliver his package. They promised to send someone out in person, but it might take till the next day. Upset about the delay he decides to grab a coffee. While in the kitchen he overhears some of his colleagues talking about a fire that just broke out in the company's experimental lab. It seems the first responders have a tough time finding survivors since they are unsure where everyone is supposed to be at the current time.

Drained from the news he decides to leave for the day at 5PM, but wished he had stayed a bit longer, because he ended up being stuck in rush-hour traffic. Later that evening he receives an email from Tom, an old college friend Bob has not seen for years. Tom tells him that he was in town for a trade show and decided to surprise Bob with a visit, but unfortunately the receptionist could not reach him and was not sure when he was going to be back in his office so he did not meet him. Disappointed that he missed Tom, he goes to bed and hopes for a better day tomorrow.

This baseline scenario highlights a number of possible application areas for which indoor location prediction can be used to improve existing or offer new services. For example it could enhance existing calendar applications by showing at the beginning of a day when an occupant is most likely in their office and have no meetings scheduled [51]. By knowing if occupants are going to be in their office within the next hour it can enable smarter building control systems capable of optimizing energy consumption based on occupants' movement patterns (*e.g.*, by

realizing zoned predictive temperature control) [62]. Such systems could minimize operational costs while also limiting the impact on occupant comfort [53]. Furthermore an indoor prediction system could enable an application that would allow us to more easily have *ad-hoc* meetings with co-workers [51] or enable smart door plates that show when the person is most likely to be back in their office [51]. This will reduce the flood of emails people receive daily and in turn save time and cost. Knowing when a person is in their office will also allow robotic assistants such as an inner-office MailBot to plan their route more effectively and thus save their limited power supply and be more accurate in its mail delivery. By combining the robot with a system that asks for help [60], MailBot can also ask occupants in nearby offices to receive the package in case the intended recipient is not in their office as scheduled. First responders can benefit from a location prediction system by knowing where missing persons are most likely located and focus their search efforts on these regions of a building. Such a system can also tell them if a person is even supposed to be in the building at the moment. Furthermore a prediction that an office worker is leaving within 30 minutes can improve notification systems such as Google Now or Siri by issuing a warning that traffic is bad and that they should leave early to arrive at their destination on time. Finally it can enable smarter automated receptionists [26] that helps visitors know when a building occupant is likely to return to her office, or arrange *ad-hoc* meetings whilst occupants are transitioning between locations [51].

Using indoor location prediction the visionary scenario would look much different. Instead of relying on reaching Alice via email, Bob's calendar application already identified possible meeting times between them in the afternoon and allows him to schedule a meeting using his voice assistant. Upon reaching his office the temperature will be at his preferred temperature since the facility management's predictive temperature control system started to heat up his office before he arrived. Coming back from his lunch meeting he noticed that the temperature is slightly below his preferred temperature. He realizes that the prediction is not perfect, but it is only a few degrees off and for the sake of the environment he is ok with being not perfectly comfortable for a few minutes. Shortly after he is back in

his office he hears a knock at the door and lets MailBot in with the package he was waiting for. Since his office neighbor is not yet back from his lunch, even though he is usually has a very routine schedule, MailBot asks Bob if it is ok to leave a letter with him for his neighbor. He agrees and MailBot sends an email notifying his neighbor about where to pickup his letter. When his colleagues letter tell him about the fire he is glad to hear that the first responders were able to find everyone very quickly, which in part is thanks to a predictive system that told them where to focus their search. Before he leaves for the day, Tom, who tells him that he tried to reach him in the morning, but he was in meetings, surprises him with a visit. The receptionist was able to tell Tom when to come back since she had a predictive schedule for Bob available. Bob decides that it would be great for Tom to see his family and he is glad for his smart notification system that tells him to leave a bit earlier than usual due to high traffic on his usual route back home. In the evening, he happily falls asleep and cannot wait for what the next day brings.

Difficulty of Predicting Indoor Location

As we saw from the illustrative example, predicting occupant indoor locations and the transition time between these locations promises to solve interesting problems, however, it is also challenging. Existing systems that monolithically approach this task, combining a prediction of where a user will go with one of when they will go there, perform poorly with prediction results that are close to the performance of a majority predictor (*i.e.*, predicting that the occupant will always stay in his most frequented location) [11,61]. This is likely because occupants' spatiotemporal routines within the interior of a building, such as a workspace, can be very complex. They can change substantially from weekday to weekday, and encompass many destinations and transitions in relatively short spans of time. One factor that makes it difficult to accurately predict an occupant's transition time into a space is the fact that indoor spatiotemporal routes are typically short (meters to hundreds of meters) and traversed rapidly (in seconds to minutes). This is in contrast to the problem of outdoor location prediction (*e.g.*, [39,76]), for which travel times and distances are long enough that a useful prediction algorithm only needs to predict the next significant location after the user has already departed. The longer transition time

allows for the use of features such as currently traversed path to make predictions, which are mostly not useful to applications of indoor prediction, due to the shorter indoor transition times. For example, when Bob leaves work, it is possible to predict where he is going next based on the partially traversed path. If he drives south he might go to the gym for his weekly workout, while if he drives east he is on his way home. An algorithm can make a decision based on this observation. Since traversing outdoor spaces usually has longer commute times it is possible to predict Bob's arrival time sufficiently ahead of time. For an indoor case it is possible to predict Bob's next location based on the observed path. However, because the traversal time is so short, by the time enough of a path is observed, Bob would already be at his target location. This would preclude many, if not all, of the applications mentioned earlier. In addition this also means that many of the analytic techniques that perform well outdoors do not work well when applied to indoor scenarios.

Given the complexity of human behavior and the difficulty of predicting a person's transition time, most prior work has focused on predicting an occupant's next location without taking into account when these transitions will take place [44,52,55,71,72]. Solving this problem alone precludes many applications that rely on knowledge about transition timings. For example, a system that proactively heats a room in advance of an occupant's arrival [38,62] would not know whether the person was coming in five minutes or two hours, and could result in either significant discomfort (if heated too late), or significant wasted energy (if heated too early). Making exact predictions about when a person transitions and to where is a very hard problem due to the variability in length of stay at the current location. How long a person stays at a given location is dependent on their spatiotemporal routine and can vary greatly from weekday to weekday and fluctuates depending on prior events that occur during a day. Considering the difficulty of this problem, even given its importance for different applications, we believe it is necessary to model a person's spatiotemporal routine and directly use it for predictions of where a person will be and when. Evidence exists [37,62] that modeling a person's spatiotemporal structure will lead to better predictions. What adds to the complexity

of modeling this structure is that it can occur over different temporal cycles. For instance, a repeating daily cycle might involve arrival at a workplace at a certain time in the morning, while a weekly or monthly cycle might involve a regular meeting slot. Event reoccurrence with a high probability can be described through features such as time of day or day of week and can be detected by most machine learning techniques. Infrequent events on the other hand can be obscured by more frequent events that usually happen at the same time and thus it is difficult to make accurate predictions for those events. For example, if we observe that a person arrives at their office at 9AM in the morning in 80% of the cases and at 10AM only in 20% of cases, most machine learning algorithms will have difficulties predicting the 10AM arrival since the 9AM arrival is much more likely. To model complex human spatiotemporal structure, this dissertation explores the use of conditional frequent-pattern trees (FP-Trees) as a representation of this structure, through an algorithm called *ABC-Pattern-Extract* (**A**ctivity, **B**ehavior and **C**ontext Sensitive **Pattern** Tree **Extract**ion). Furthermore it explores the use of FP-Trees for predicting future spatiotemporal behavior, through an algorithm called *ABC-Pattern-Predict* (**A**ctivity, **B**ehavior and **C**ontext Sensitive **Pattern** Tree **Predict**ion).

Efficient Room Temperature Control

Indoor location prediction allows us to realize a multitude of different applications, some of which are highlighted in the introductory scenario. The focus of this dissertation are the pattern extraction and prediction algorithms, ABC-Pattern-Extract and ABC-Pattern-Predict. To evaluate the algorithms we chose one application domain, efficient office temperature control, since the infrastructure that allowed us to collect the necessary ground truth and training data also enables us to run detailed experiments without the need for the creation of additional test infrastructure. Evaluating other application domains would require us to develop addition test infrastructure. In addition, efficient room temperature control allows us to evaluate both False Positive and False Negative errors since the first impacts energy consumption and the second comfort. This means we can see how the algorithm reacts in a real-world scenario for different error cases. This exploration of error conditions, different applications (energy consumption and comfort) within our domain, and the generality of our approach lead us to believe that our

approach can be applied to multiple domains. Our evaluation shows the impact of our algorithms on the errors as well as the domain and we believe that given the errors made by the prediction algorithm it is also possible to realize other applications. As we will see later, our algorithm especially improves the impact on the human user.

Analyses by the U.S. Department of Energy shows that buildings make up for nearly 41% of primary energy usage in the U.S., with commercial buildings contributing half of that. 40% of the energy use [70] in commercial buildings is due to Heating, Ventilation, and Cooling (HVAC) systems. While modern HVAC systems make use of Variable Air Volumes (VAV) units for independent control of thermal zones [31], most modern buildings still use static schedules to run HVAC systems, thereby wasting energy when spaces are unoccupied [8,19,20,24,74,75]. Thus, recent work has focused on actuating HVAC systems, based on near real-time occupancy information [8,19], as well as predicted occupancy-based on learned patterns [37,62], with estimated HVAC energy savings ranging from 30-40% in the best case [19,62].

Given the high potential for positive environmental, economic, as well as human impact, predictive occupancy-based office temperature control is an attractive target to test the pattern extraction and prediction algorithms. A successful predictive algorithm for this application domain is not only reducing the energy consumption of temperature control systems, but also maintaining or even improving human thermal comfort. To investigate the potential impact of predictive temperature control we first present a simulation framework that quantifies the energy consumption and comfort impact of predictive temperature control under varying levels of false negative and false positive error. To evaluate the framework we performed an EnergyPlus consumption analysis using 235 office occupancy traces collected over a period of six months. The framework reveals that predictive algorithms can save up to 16.88% over a reactive temperature control system even for high error cases with a 25% False Negative and False Positive rate. Motivated by the high potential for savings, ABC-Pattern-Predict is tested on a four month, 53 offices occupancy dataset. It achieves a very high accuracy of 92.56% and an

increase in recall by 10.93% over the current state of the art, PreHeat. The high accuracy and recall improvement over PreHeat leads to an estimated energy consumption reduction of 16.1%, while maintaining the same level of human thermal comfort.

Research Questions and Statement

Through this dissertation we aim to answer the following question: **Given complexity of modeling human spatiotemporal behavior and reoccurring events how can we compute conditional frequent-pattern trees for both in a space and time efficient manner (Q1)**. Human spatiotemporal behavior is influenced not only by the current situation (such as current location and time of day), but also larger contextual factors such as the temporal configuration (*e.g.*, current day of the week, day of month, etc.) or previous locations and events. Given this variability it is important to define a descriptive model that is flexible enough to capture the complex structure found in human behavior and extensive enough to adapt to new situations and data. The second central question for this dissertation is, **how well does the detected structure in a person's spatiotemporal behavior allow us to make predictions for future events (Q2)**. Human behavior is governed by structure that is either imposed on us externally, for example, through reoccurring meetings we have to attend or it happens internally through our choices and subconsciously through habit formation. Through knowledge of this structure, it is possible to make predictions about future spatiotemporal events. Lastly we aim to answer the question **how do the characteristics of the prediction results impact our ability to realize the above-mentioned applications (Q3)**. Many of these applications react differently to prediction errors. By understanding the errors we will be able to evaluate the feasibility of the pattern-based prediction system for different classes of applications. This will also allow us to adapt the applications to the errors that our prediction approach makes. For example if we consider predictive temperature control and we know that the prediction mostly predicts that a person is not at a location even if they are (*i.e.*, a false negative), then this will have implications for occupant comfort. One way to address this error is by limiting the deviation from

the occupant's preferred temperature and thereby change how fast the room reaches this temperature.

All of these questions lead to the central statement for the dissertation:

Conditional frequent-pattern trees offer a flexible framework for modeling and understanding of a person's spatiotemporal behavior. Using these trees to find reoccurring structure in that behavior, allows us to improve the predictive performance of spatiotemporal events and support diverse applications.

This dissertation makes the following contributions:

1. It presents an algorithm, ABC-Pattern-Extract, capable of explicitly modeling the complex human indoor spatiotemporal structure in a time and space efficient manner, so it can run on a standard laptop without high-level parallelization (*e.g.*, Hadoop) taking less than an hour per user. Doing so will allow us to predict future spatiotemporal events and in turn enable the realization of aforementioned applications.
2. Through ABC-Pattern-Predict it shows how the modeled structure can be used to accurately predict future spatiotemporal events. The results show that it is more accurate than current state of the art algorithms and also greatly improves on the recall error metric.
3. To show the impact of the algorithm on an application domain this dissertation uses office temperature control as an example. Compared to previous state of the art algorithms, ABC-Pattern-Predict improves the potential for energy savings by 16.1% over previous state of the art techniques, while maintaining the overall human thermal comfort.

Dissertation Structure

In order to explore the central dissertation statement, this document, in Chapters 2 and 3, gives an overview of prior indoor location prediction algorithms and additionally highlights the diverse evaluation measures needed to thoroughly evaluate prediction algorithms. We introduce a number of prediction algorithms

and re-implemented three of the most accurate, and test them on a variety of datasets. The results show the importance of diverse evaluation measures.

Secondly, in Chapter 4, we show the usefulness of different spatiotemporal features by introducing an indoor location prediction algorithm, Indoor-ALPS. Our algorithm uses various spatiotemporal features to make two separate predictions: when is a person transitioning and where are they transitioning to. Our results on a publicly available dataset shows a very high predictive performance, which we evaluated using the metrics introduced in Chapter 2.

Chapter 5 describes the central pattern extraction and prediction algorithm, *ABC-Pattern-Extract* (**A**ctivity, **B**ehavior and **C**ontext Sensitive **Pattern** Tree **Extract**ion) and *ABC-Pattern-Predict* (**A**ctivity, **B**ehavior and **C**ontext Sensitive **Pattern** Tree **Predict**ion). Using lessons learned from Indoor-ALPS, we created an improved algorithm that directly models frequent and infrequent spatiotemporal structure. Our algorithm allows us to make predictions of future spatiotemporal events.

Chapter 6 shows the results of our algorithm using the measures introduced in Chapter 2. In addition we highlight the usefulness for efficient office temperature control, one of the applications described in our scenarios above. Over 40% of a building's energy consumption is due to temperature control, which makes this an attractive target for optimization. As we can see in Chapter 6 there is great potential for using predictive temperature control for energy consumption reduction. Our results show a very high algorithmic performance with an accuracy of 92.56% and recall of 79.77%. We were also able to show a 10.93% improvement in Recall over PreHeat, the best state of the art algorithm. An energy simulation shows potential energy savings of 16.1% over PreHeat, while maintaining the same level of occupant comfort.

Chapters 7 and 8 discuss the results and show potential extensions and future work.

2. Measures Used in Document

In order to evaluate the performance of prediction algorithms a number of statistical measures were used in the past. We believe that a comprehensive reporting using multiple measures is necessary to evaluate the true performance of an algorithm. In the following section introduces six different measures that we believe are important for the evaluation of an algorithm and are used for the remainder of this document.

2.1.1. Accuracy

The accuracy of a machine-learning algorithm is the most commonly reported statistic and describes the proportion of correct predictions from the total number of predictions made. Since datasets are rarely uniform (*i.e.*, they vary in number of data points and number of locations per user) the accuracy is best calculated over the whole data set irrespective of individual users or locations. Calculating the accuracy on a by user or location basis and reporting on mean and standard deviation can prove difficult, because of differences in number of data points per user in a dataset. For example a dataset might have users with as few as six data points or others with over 1000 data points; comparing users with diverging number of data points as the same would lead to a misleading result. Even though accuracy is a standard metric it can deliver a misleading picture of the algorithm performance, because it is highly influenced by the underlying location distribution. For this reason additional metrics are required.

2.1.2. Precision and Recall

Precision and recall are information retrieval concepts that provide an understanding and measure of relevance. For a given class (or result), precision describes the proportion of data points that belong to class from the full set of those that are predicted to do so. Recall, on the other hand, describes the ratio of relevant instances that are actually retrieved. Both precision and recall are calculated on a per class basis for each entity in the data set. The overall precision

and recall for the algorithm on a specific data set is calculated using a weighted average (on the occurrence frequency) of the class specific precision and recall.

Precision and recall are calculated in terms of true positive, false positive, and false negative. For a given class, true positive (TP) is the number of data points that are correctly predicted to belong to that class, false positive (FP) is the number of data points that are wrongly predicted to belong to that class, and false negative (FN) is the number of data points that belong to a class, but are not predicted to do so. Based on these values precision and recall are mathematically defined as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Very often Precision and Recall are expressed through the F1-Score also called the harmonic mean of precision and recall. It is mathematically defined as:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

2.1.3. Kappa Statistic

The Kappa statistic describes the class-wise agreement between actual data and predicted data. It is determined by first calculating the number of data points correctly predicted to be in each class and the number incorrectly predicted to be in any other classes. This generates a confusion matrix of the form illustrated in Table 2.

Table 1: (Adapted from Congalton and Green 1999, p. 47)

i = Predicted Classes (classifier outputs)	j = Observed classes (labeled reference data)			
	j_1	j_2	j_k	n_{i*}(row totals)
i_1	n_{11}	n_{12}	n_{1k}	n_{1*}
i_2	n_{21}	n_{22}	n_{2k}	n_{2*}
i_k	n_{k1}	n_{k2}	n_{kk}	n_{k*}

n_{*j}	n_{*1}	n_{*2}	n_{*k}	$n_{**}=n$

(column totals)

The elements on the top-left to bottom-right diagonal show the number of data points in which the predicted data agrees with the observed data. Other cells show misclassifications. For example the entry n_{12} depicts the number of data points that belong to class j_2, but which are classified as class i_1. Column ($n_{i\square}$) and row means ($n_{\square j}$) are then generated from this table. Finally, Kappa provides an aggregate measure of agreement, ranging between -1 and 1, between the observed and predicted values. It is calculated in the following way:

$$Estimated\ Kappa = \frac{n\sum_{i=1}^{k} n_{ii} - \sum_{i=1}^{k} n_{i*}\, n_{*i}}{n^2 - \sum_{i=1}^{k} n_{i*}\, n_{*i}}$$

There are different schemes for interpreting Kappa values. We use the one introduced by Landis and Koch [42] and described in Table 3.

Table 2. Interpretations of Kappa introduced by Landis and Koch [42].

Kappa	< 0	0.01 – 0.20	0.21 – 0.40	0.41 – 0.60	0.61 – 0.80	0.81 – 0.99
Agreement	Less than chance	Slight	Fair	Moderate	Substantial	Almost perfect

2.1.4. 0-R Accuracy

The 0-R accuracy, or baseline predictor, always returns a prediction of most the frequently occurring location in a dataset. The accuracy of the baseline predictor shows if one location, or possible result, dominates other locations. For example, a 0-R accuracy of 87.5% would indicate that the data set features 87.5% of cases that belong to a single location. The 0-R accuracy also serves as a useful way of assessing performance improvements – an algorithm that leads to 90% accuracy when the baseline is 87.5% offers only a modest level of predictive power, while one that offers 80% when the baseline is 40% may be, practically, a much more powerful and useful system. However if the baseline accuracy is already very high

(> 90%) and an algorithm still offers consistent improvements using the same algorithm we can expect similar if not greater improvements on datasets with lower baseline accuracy.

3. State of the Art in Indoor Location Prediction

The motivating scenarios showed how indoor location prediction could be used for various application areas. These applications do not only rely on where a person is going, but also when they will arrive at a location. Even though predicting a person's transition is vital for a number of interesting applications most prior literature focused on predicting a person's next location. Investigating further, an analysis of prior research showed that indoor location prediction algorithms can be broadly classified into three categories: next location prediction irrespective of time [44,52,55,71,73], next location and transition time prediction [11,37,61], and room-level occupancy prediction [62].

Petzold *et al.* [55] presented an early and influential analysis of indoor location prediction algorithms. They predominantly focus on the computational cost and time/space complexity of the algorithms and assess performance simply in terms of accuracy. They note this figure remained at approximately 80% across the set of algorithms they surveyed. Voigtmann and David [72] (P10) present a more recent survey of the field and they re-implemented and re-evaluated selected algorithms. Again the analysis concentrated on conceptual aspects such as the nature of the data set (real, simulated or self-collected, etc.) and whether a comparison to other approaches was documented. Once again, they reported the mean accuracy for each selected algorithm on each data set, but provide no further characterization of performance.

Given the seminal nature of Petzold *et al.* [55] in the field, we choose to select algorithms for our research by tracking and examining all papers that cite it. Articles from this set were retained for analysis and discussion only in cases where they introduce a new indoor location prediction algorithm, tested existing algorithms on new data sets, or modified and improved on existing algorithms. This

led to a final set of nine articles, which are summarized in Table 1. Since it is necessary to evaluate algorithms using a wide range of measures (see Chapter 2), we also highlight the limited way in which performance of these algorithms has been evaluated: seven out of nine were tested on a single dataset and eight out of ten report performance only in terms of accuracy.

Table 3: Reviewed Indoor Location Prediction papers. A check mark indicated if a particular measure was reported or not.

Paper	Title	Accuracy	Baseline	Precision + Recall	F1-Score	Kappa	Dataset Description	Number of Datasets	Included: D/N/W	Task
P01	*Comparison of Different Methods for Next Location [55]*	✓	✗	✗	✗	✗	(✓)	1		Next-Location Irrespective of Time
P02	*A Dynamic Bayesian Network Approach to Location Prediction in Ubiquitous Computing Environments [44]*	✓	✗	✗	✗	✗	✓	1		Next-Location Irrespective of Time
P03	*Using an Adaptive Search Tree to Predict User Location [52]*	✓	✗	✗	✗	✗	✓	1		Next-Location Irrespective of Time
P04	*A predictive location-aware algorithm for dementia care [73]*	✓	✗	✗	✗	✗	✓	1		Next-Location Irrespective of Time
P05	*Person movement prediction using neural networks [71]*	✓	✗	✗	✗	✗	✓	1		Next-Location Irrespective of Time
P06	*Occupant Location Prediction Using Association Rule Mining [61]*	✓	✗	✗	✗	✗	✓	2	Day-time analysis	Next-Location Given Time
P07	*Predicting Future Locations and Arrival Times of Individuals [11]*	✓	✗	✗	✗	✗	✓	1		Next-Location Given Time
P08	*PreHeat: Controlling Home Heating Using Occupancy Prediction [10]*	✓	✓	(✓)	✗	✗	✓	1		Occupancy Prediction
P09	*A Survey To Location-Based Context Prediction [72]*	✓	✗	✗	✗	✗	✓	4		Survey

3.1. Prediction Tasks

Indoor location prediction is not a homogenous task and research has tended to focus on three specific problems: Next Location Prediction Irrespective of Time; Location and Transition Time Prediction and; Room Occupancy Prediction. The following sections describe research on each of these problems in more detail.

3.1.1. Next Location Prediction Irrespective of Time

Perhaps the most common indoor location prediction task is to determine the next location that an individual will visit independent of the time this movement will take place. Such systems attempt to answer the question "where are you going next?" and typically rely on a user's current location and location history in order to answer it. One of the most popular data sets used in attempts to solve this problem is the Augsburg Indoor Location Tracking Benchmark. This dataset contains the location transition traces for four office workers as they move around one floor of a large university office building over a seven-week period. A more detailed description of the dataset can be found in Chapter 4. In early work using this benchmark, Petzold e*t al.* [55] (P01) implemented and analyzed seven different algorithms for next place prediction: Bayesian networks, multi-layer perceptron, Elman Net, Markov predictor, state predictor, Markov predictor with confidence counter, and state predictor with confidence counter. Their results showed that an accuracy of up to 79.68% could be achieved using the Elman Net. They also compared the algorithms on criteria such as accuracy, quantity (the fraction of requested and returned predictions), learning and relearning speed, and computational and modeling costs, concluding that the Markov and State Predictors have the lowest modeling and computational costs.

Vuong *et al.* [73] (P04) introduced the *Adaptive Confidence Estimator* for next location prediction a technique that improved upon Petzold's state predictor [14]. They tested their algorithm on the Augsburg Indoor Location Benchmark and reported an average accuracy of 88.57% and quantity of 76.03%, improvements of 6.69% in accuracy and 1.65% in quantity over Petzold's original implementation. Also using the same dataset, Vintan *et al.* [71] describe a neural network based prediction approach that led to a relatively overall low prediction accuracy of 76.4% and improved performance (up to 92.3%) on specialized tasks such as determining if an individual's next destination was their office. Similarly, Oh [52] used an adaptive search tree and showed an 80.67% accuracy that was consistent across all four users in the Augsburg dataset. Compared to other papers we reviewed Oh's work provide a more comprehensive algorithm validation as it reports additional details on the dataset and analysis such as the number of data points

per user, the number of weeks for each user in each season (summer or fall), or the time complexity of the algorithm.

Other authors who have examined next location prediction have done so using custom datasets. For example Lee *et al.* [44] (P02) explored the use of Bayesian networks for next location prediction based on a dataset derived from a diary study of 366 undergrad students over a period of two days. In this study, participants noted down the start and end time of "actions" or activities, the location at which these were performed, and the route they had taken to arrive at the location. From the 672 days of data they collected, 266 days were retained for analysis (days with very short times and impossible routes were excluded) and the performance of four predictive algorithms assessed. They report that a Dynamic Bayesian Network led to the highest accuracy (72.67%), but as the data set is not publicly available and there is only a limited description and discussion of algorithm performance it is hard to draw further concrete conclusions.

As many of the algorithms tackling the next location prediction problem use the Augsburg data set, it is trivial to compare amongst these based on accuracy, the most commonly reported metric, and conclude that the *Adaptive Confidence Estimator* [73] offers currently optimal performance. However, many aspects of the algorithms and Augsburg data set remain currently unexplored. For example, the current literature does not report precision, recall, 0-R accuracy or Kappa and, as such, we do not know how well different algorithms perform on different classes or locations. Furthermore, there is little explicit discussion regarding the distribution of classes within the dataset as a whole. Indeed, given the dominance of this dataset in the literature, we highlight the current lack of critical evaluation of its suitability as a basis for developing predictive algorithms as a particular area of concern.

3.1.2. Location and Transition Time Prediction

This task involves predicting a user's location at a specific time in the future given a time-stamped movement history. By executing a series of these predictions with increasingly long durations, this method can also predict a transition time between locations. In other words, not just where an individual will be going next, but the time at which they will do this. Compared to next location prediction, there are

relatively few systems that attempt this task, possible because it is more challenging – it involves predicting spatiotemporal patterns, rather than simply spatial patterns. .

Work on this problem has also used the Augsburg data set. For example, Ryan and Brown [61] (P07) developed an Association Rule Mining approach. To test their system they used self-recorded location traces from five office workers, generated a synthetic data set, and used the Augsburg Indoor Location Tracking Benchmark as an external evaluation. They report an accuracy of up to 79% when predicting a user's location after one hour on the data set they collected themselves. However, on the Augsburg data they achieved an accuracy of up to 56%, a drop they explained due to the greater variability of movements and routines in the Augsburg data. Ryan and Brown did not report measures such as 0-R accuracy or Kappa, but did analyze the performance of their algorithm for each hour of the day. Their by-hour analysis showed that the accuracy is higher in the beginning and end of the day and drops notably during the middle of the day. Furthermore, by testing the algorithm on three different datasets the authors provide a more comprehensive characterization of performance, highlighting situations in which it is more or less successful. However, as the authors focus solely on accuracy this picture remains incomplete and it is impossible to determine the specific situations or ways in which the algorithm fails. For example, it is unclear if the algorithm is consistently predicting that the user arrives at a location earlier than in reality. Association rule mining additionally has the drawback of not being able to consider the order of events. For many time series problems the order is an important source for structure and predictability, which makes it vital that such algorithms

Burbey developed Prediction by Partial Match (PPM) to solve the problem of predicting a location given a time [11]. This system was tested on the publicly available UCSD Wireless Topology Discovery dataset, which contains WiFi-based indoor location tracking traces for 275 freshmen students over a period of 11 weeks. PPM relies on the majority location for a given timeslot an achieved an impressive mean accuracy of 87%. However, once again, measures such as 0-R

accuracy or Kappa were not presented. Based on the Burbey's explanation, we infer PPM relies on users exhibiting a highly regular spatiotemporal routine in order to achieve a high accuracy but that this is unlikely to be maintained in cases where users behave more erratically. However, based on the current literature, PPM represents a best of class approach to the problem of predicting an individual's location at a specific time.

3.1.3. Occupancy of a Room

The final task involves predicting whether or not a given location (typically a room) will be occupied or not at a specific time. The movements of individuals are not explicit considered in this task and predictions are made based on occupancy logs for locations. As it focuses on places rather than users, algorithms that solve this type of prediction problem are particularly relevant for infrastructure and building operations systems.

In some of the earliest work on this topic, Scott *et al.* [62] (P09) developed PreHeat, an occupancy prediction algorithm for homes. They collected data from five households and used their algorithm to make predictions about when the entire home and the individual rooms would be occupied. Their results show a median prediction accuracy of above 80% and they also present an ROC curve highlighting the tradeoff between misses (failing to detect occupancy) and false alarms (incorrectly detecting occupancy). They do not discuss class-wise performance measures such as Kappa or the 0-R baseline. The algorithm relies on a binary majority vote (occupied or not occupied) of data from the five previous days that most closely resemble occupancy patterns of the current day. By basing the prediction on a spread of similar days the algorithm implicitly models the spatiotemporal routine of a person and achieves a relatively high level of performance. However, its weakness are that it does not attempt to model more complex spatiotemporal structure based on contextual factors such as the length of previous occupancy or the specific day of the week. The next subsection will give a detailed description of the algorithm.

3.2. Algorithm Description

In order to get a better understanding of the current state of the art, we chose one algorithm for each prediction task that showed the highest reported accuracy and was also tested on a real world and not simulated data set for reimplementation. For next location prediction, we chose the *Adaptive Confidence Estimator* [73]. For predicting the location at a given time, we chose *Prediction by Partial Match* algorithm [11]. For occupancy prediction, we chose *PreHeat* [62].

For the remainder of this section we give a detailed description of each selected algorithm, explain what kind of data set structure is exploited by the algorithm, and what limitations we found on each data set.

3.2.1. Adaptive Confidence Estimator (Next Location Prediction)

Accuracy	*Baseline*	*Prec. + Recall*	*F1-Score*	*Kappa*	*Real-World Dataset Used*	*Number of Datasets*
✓	✗	✗	✗	✗	✓	1

Table 4 ACE Metric Summary

Vuong *et al.* [73] introduced the Adaptive Confidence Estimator (ACE) as an extension to the state predictor with confidence counter technique used by Petzhold *et al.* [56]. The algorithm makes its predictions based on the last *n* visited locations and uses a state machine to predict the location following this sequence of visited locations. The author's argue that for some applications it is beneficial to withhold predictions that have a high probability of being incorrect, which is why the algorithm also includes a measure of how confident it is with the predicted location. Low-confidence predictions are not returned by the algorithm, which leads to the situation that only a subset of the prediction requests are satisfied.

Pattern	2-state	Confidence Counter
…	…	…
$p_1 \ldots p_n$	*L1*	*s*
…	…	…

Table 5 ACE History Look-up Table

The algorithm is implemented through a history look-up table (see Table 2). Each entry in the table consists of a pattern, which is a collection of the last *n* consecutively visited locations, an associated state (L0 or L1) for the state machine, and the current confidence counter that determines if the confidence in the result is high or low. The order of a state predictor describes the length of a pattern and thus a state predictor of length *n* uses the last *n* locations. The current state of the state machine determines the next location following a pattern p. After each prediction query this state is being updated using the underlying state machine for the corresponding pattern. An example of such a state machine for a pattern p of visited locations is shown in Figure 1.

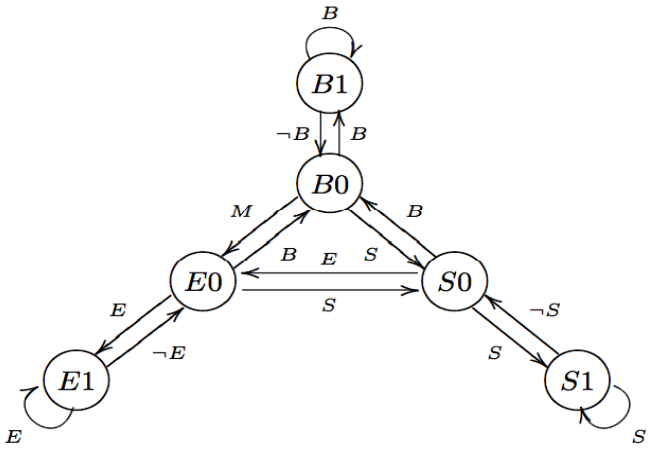

Figure 1 Representation of ACE State Machine (adopted from [56])

Corresponding to this state machine, three possible next locations (*i.e.*, B, E, and S) following the pattern *p* can be observed. Each location has two possible states 0 and 1 and we can see that the state machine only transitions to a new location if the current state level is 0. The transition within the machine is initiated by correct or incorrect predictions. For example if the state machine is in state B1 and B is the

correctly predicted next location the state would not change, if on the other hand B would be the wrong prediction the state machine will transition into B0. The next time the pattern p occurs the algorithm still predicts B as the next location. If this prediction is correct the state machine will transition back into B1. If the prediction is incorrect and the actual next location is E the state machine will transition into E0. This example shows that the state machine approach needs two examples of a new next location to change its prediction outcome. We can understand the state level of the state machine as the retraining rate for the algorithm, i.e., the rate of how many examples of a new routine need to be observed before this new routine becomes the predicted outcome of a pattern.

The confidence counter is handled independent from the state machine. The confidence counter *cc* is increased for each correct prediction and decreased for each wrong one. If the confidence counter for a pattern/current-state pair falls below a chosen threshold t the algorithm will withhold the next result of a location prediction for that pair. The confidence counter and the state machine will be updated even if the current prediction result is being withheld. The confidence counter can never fall below 0. The counter cc is increased or decreased by the frequency of the next location. This change is a deviation to the original technique described in [56] and was introduced by Vuong et al. [73] to make the predictor more robust against unexpected events. Their rational is that an unexpected deviation from a frequent event should not be penalized as strongly as a deviation from a sporadic event.

For each prediction query, a location pattern, the state predictor first checks if the pattern was encountered before. In case it was not encountered, the pattern, the next location following the pattern (as soon as it is available), an initial confidence counter of s, and the confidence threshold *t* will be added to the look-up table. No prediction is returned in that case. If the pattern was encountered before, the predictor checks the confidence counter: if it is above the confidence threshold *t* it will return the next location (the current state of the state machine) and if it is below the threshold the next location prediction will be withheld. The rationale behind the latter decision is that it is better to omit a potentially wrong result with low

confidence than return a wrong result. As soon as the next location is available the confidence counter *cc* will be updated as described in the previous paragraph. Because some of the prediction queries have potentially no result (in cases of new patterns or patterns with low confidence) only a percentage of the queries will be processed and thus an additional variable is needed for ACE: quantity. The quantity of a prediction represents the percentage of answered prediction queries vs. total prediction queries.

The Adaptive Confidence Estimator is based on the assumption that people follow a predictable routine of visited places. For example if we know that a user visited place A and place B we expect to see place C in most cases. The ACE algorithm is very accurate with low errors if and only if a person is following the same routine independent of time or day of week/month. It does not take any temporal information into account. As Table 4 highlights again, the results reported by Vuong et al. [73] also do not highlight what kinds of errors the algorithm makes or how well the algorithm predicts non-frequent locations. For the former it is necessary to report on precision and recall and for the latter the Kappa statistic is needed.

Table 6 Prediction by Partial Match Example

Order k =2					Order k = 1					Order k = 0					Order k = -1				
Predictions			c	p	Predictions			c	p	Predictions			c	p	Predictions			c	p
ab	è	r	2	⅔	a	è	b	2	2/7	è	a	5	5/16		è	A	1	1/|A|	
	è	*Esc*	1	⅓		è	c	1	1/7	è	b	2	2/16						
						è	d	1	1/7	è	c	1	1/16						
ac	è	a	1	½		è	*Esc*	3	3/7	è	d	1	1/16						
	è	*Esc*	1	½						è	r	2	2/16						
					b	è	r	2	⅔	è	*Esc*	5	5/16						
ad		a	1	½			*Esc*	1	⅓										
		Esc	1	½															
					c		a	1	½										
br		a	2	⅔			*Esc*	1	½										

		Esc	1	⅓				
					d	a	1	½
ca		d	1	½		*Esc*	1	½
		Esc	1	½				
					r	a	2	⅔
da	è	b	1	½		*Esc*	1	⅓
		Esc	1	½				
ra		c	1	½				
		Esc	1	½				

3.2.2. Prediction by Partial Match (Location Prediction for a given time)

Accuracy	*Baseline*	*Prec. + Recall*	*F1-Score*	*Kappa*	*Real-World Dataset Used*	*Number of Datasets*
✓	✗	✗	✗	✗	✓	1

Table 7 PPM Metric Summary

Ingrid Burbey [11] introduced Prediction by Partial Match (PPM) as a method for indoor location prediction. The original technique was developed by Cleary and Witten [13] as a text compression tool.

The PPM algorithm uses an r-order Markov Model to predict the symbol that follows a sequence of symbols. For a given input training string of length *l* it breaks up the string into substrings of length *t* (*1* ≤ *t* ≤ *r* and r < l). The order of symbols in the substring is preserved from the original string. For example the substring of length 2 for the string *abra* are *ab, br,* and *ra*, but not *rb*. For each possible substring the model records all possible symbols that follow that substring. Attached to each entry in the model is a count that records the frequency of occurrence of that substring/next-symbol combination. Each substring/next-symbol entry has an associated probability that is calculated as the entry's frequency count over the total number of instances of the entry's substring. An example for the 2nd order model for the string *abracadabra* is given in Figure x.

To use the above-described model for location prediction the input data is formatted into a daily, ordered sequence of time/location pairs. Several weeks of training data is then used to build a trained model for prediction. Throughout the algorithm a 1st order Markov model was used with the rationale that knowing where someone was shortly before the current location is not giving any additional useful information. The time component in the time/location pairs is slotted into 10-minute chunks and only significant locations are combined with these chunks. The definition by Ashbrook and Starner [4] for significant location was used to determine if a given location is significant or not.

During the validation phase the location for each time/location pair in the test data set was predicted by checking the trained model for that particular time slot. If the time slot was encountered before and subsequently appears in the model the location with the highest probability was returned. By comparison to the actual encountered location a decision was made if the prediction was correct or not. If no prior data is available for that particular time slot the most frequent location was returned.

Prediction by Partial Match relies on a person always visiting a place at exactly the same time. If that holds true the algorithm will be very accurate. Unfortunately humans are rarely that predictable and the arrival at a place depends on the time of day, day of week, or previously visited places. For example people might follow a set routine and even though the routine starts later and every part of the routine is shifted in time it is still predictable. PPM would not be able to account for this since it is expecting a routine to always happen during the same time frame. As we can see in Table 7 the original work also did not report on additional measures such as precision/recall, Kappa statistic, or baseline accuracy.

3.2.3. PreHeat (Occupancy Prediction)

Accuracy	*Baseline*	*Prec. + Recall*	*F1-Score*	*Kappa*	*Real-World Dataset Used*	*Number of Datasets*
✓	✓	(✓)	✗	✗	✓	1

Table 8 PreHeat Metric Summary

Scott et al. [10] introduced the PreHeat algorithm as part of their effort to predict occupancy for residential spaces. We believe that this algorithm is also highly applicable to an office scenario even though the usage patterns of offices differ from homes.

The PreHeat algorithm is based on the intuition that people follow the same patterns in their life and thus by observing a partial pattern it is possible to predict the remainder of a pattern. For this algorithm the data sets are formatted into room-specific 5-minute time slots and for each timeslot a binary indicator represents if the associated room is occupied during that time slot or not. Essentially for each day a binary, ordered vector of length 288 is created and each entry in the vector represents the occupancy of the room during that 5-minute time slot that day. For each day in the test data set the algorithm tries to find the 5 closest matches to the current day by calculating the Hamming distance between the currently observed partial day vector and the day vectors that came before the current one. For example if the day up until 9:45 am is observed the algorithm calculates the Hamming distance for the partial vector of length 120 (position 0 to 119) for each day that came before the query day. The 5 days that have the smallest distance to the query day are chosen.

To calculate the probability that a room is occupied at a certain time slot in the future (I chose 1 hour into the future from the currently observed time slot) the algorithm adds up the binary occupancy for this time slot over all 5 representative days. This sum divided by 5 results in the probability of occupancy. If the probability is above 0.5 (the room was primarily occupied over the 5 days) 1 or occupied will be chosen as prediction result otherwise 0 or unoccupied will be chosen as prediction result.

The Augsburg and UCSD data set needed to be specially prepared for the PreHeat algorithm, because each of the data sets was movement-based by user and not occupancy-based by room. To get the data into the right format a new data set was extracted from the original data by calculating the occupancy for each room and day. We wrote an algorithm that checked for each possible room and each possible day if any of the users occupied that particular room during a given timeslot. If even one user occupied the room the algorithm marked the room as occupied during that time slot and day. Many of the rooms were not occupied for most of the time, which influenced the prediction results. The effects of this are discussed in the Prediction Results section.

PreHeat exploits the spatiotemporal routine of occupants to make predictions about the future. It is build on the assumption that human behavior is to a degree predictable, i.e., if the beginning of a day is similar to a previously encountered day than it is also likely that the remainder of the day is similar to that day. It follows that PreHeat will be very accurate if a person's spatiotemporal routine can be expressed by only the visited location and time of day. If on the other the spatiotemporal structure is more complex (*e.g.*, it depends on day of the week, events the previous day, other contextual events such as outside weather, etc.) PreHeat will have a hard time making predictions. Besides reporting accuracy the authors also reported on the baseline, as well as Precision and Recall (alas is a slightly modified form to make it more applicable to their application domain). We can see in Table 8 though that they did not report on additional metrics such as F1-Score or Kappa statistic.

3.3. Evaluation of Prior Algorithms

To comprehensively evaluate different prediction algorithms we believe it is necessary to apply them to different data sets and compare the results using standardized measures. Throughout this section we describe the data sets used for our analysis. For each data set we report: the number of users, the extensiveness of the data set meaning how many data points were recorded per user and day and over how many days was the data collected, how was the data collected, what was the collection frequency, and how variable was the data from user to user.

For our analysis we considered four different data sets: two publicly available data sets that were used for most of the current literature on indoor location prediction and a self-collected data set for the occupancy prediction task since the baseline accuracy on the other two data sets for that task was too high. High baseline accuracy is problematic for the evaluation of an algorithm since even the most basic algorithm already achieves a very high performance and thus making it difficult to evaluate the strength and weaknesses of an algorithm. The two publicly available data sets are the Augsburg Indoor Location Tracking Benchmark and the UCSD Wireless Topology Discovery data set. Additionally we collected a data set within a big university campus office building that captured the occupancy of each room within the building.

The datasets fall into three categories: transitional datasets, only a person's transition from one space to the next is recorded, tracking datasets, a person's movements through a space are tracked, and room occupancy data sets, not a person, but individual rooms are monitored. To fully understand the results of an algorithm we believe it is also important to explore the structure of a dataset that aided or hindered the algorithm. For example, including evening and night in the analysis may bias the results, as many office buildings are unoccupied during these periods. To that end we also report additional analysis for the transitional and tracking data set that shows how much of each data set consists out of the two most frequent locations and for the occupancy data set we report the average

occupancy of the rooms during the whole day Monday through Sunday and 8am-6pm Monday through Friday.

The high-level details for each data set are shown in Table 9.

Table 9 Dataset Characteristics

	Augsburg Data Set	UCSD Data Set	Office Occupancy Data Set
Tracked Entity	Person	Person	Room
No. of entities	4	275	265
Total no. of locations	15	200	265
Avg. no. of days	23.25 (SD 3.42)	25.56 (SD 20.81)	188.65 (SD 4.3)
Avg. no. of data points per entity/day	316.60 (SD 8.55)	508.46 (SD 739.1)	292.57 (SD 0.84)
Avg. no. of locations per entity/day	6.70 (SD 0.70)	3.45 (SD 2.68)	1 (SD 0)
Collection Method	Smart Door Plate	PDA-based WiFi triangulation	BACNet Sensor Queries
Collection Frequency	Every Room Change	Every 20 sec	Every 10 minutes
Flaws	Summer and Fall data sets 4 month apart		
Room vs. Floor vs. Campus	Covers 1 floor in an office building	Covers several campus buildings	Covers every floor in one university office building
Type	Transitional	Tracking	Occupancy

3.4. Augsburg Indoor Location Tracking Benchmark

The University of Augsburg through summer and fall 2003 as a benchmark collected the Augsburg data set for their Context Prediction Project [66]. The data was collected using a smart doorplate concept: each user was tracked through an RFID card upon entering of a space (office, corridor, kitchen, etc.). The authors tracked four individuals who worked on the same floor for an average of 6.75

weeks (SD=1.71). Due to the smart doorplate only transitional data from one space to another is available.

Note that the summer and fall data sets are nearly four month apart. Since the data was collected from university office workers that rarely have a consistent routine over such a long time we concentrated our analysis on the fall portion of the data set. The fall data set consists on average out of 5 weeks (SD=1.41) of data. In addition several papers that used this data set for testing reported the highest accuracy on user A of the data set (some methods differ by up to 15% between user A and the other 3 user) even though the least amount of data is available for this particular user, with only 1 week of data during summer and 4 weeks of data during fall.

Figure 1 shows an analysis of how much data is covered by the two most frequent locations (Top 1 and Top 2). We calculated the values individually for each user and report on the mean and standard deviation across all users. The data is split into all-day and 8am to 6pm, because we believe it is important to analyze the working hours separately since most offices are empty during the night. We can see that the top two most frequent locations on average over all users cover 71.62% (SD=1.65%) of the data all day and 77.74% (SD=4.81%) of the data during daytime. It is important to note that the most frequent location (Top 1) is on average only under 50% present in the data set. This means that a 0-R prediction algorithm that always predicts the most frequent location cannot achieve accuracy higher than 50%.

3.5. UCSD Wireless Topology Discovery Data Set

The University of California, San Diego, collected this data set as part of their Wireless Topology Discovery project throughout 2002. McNett et al. [49] provided 275 UCSD freshmen with instrumented PDA's running Windows Pocket PC. They recorded more than 13 Million data points over a period of 11 weeks. The PDA's collected data every 20 seconds and recorded the 3 primary access points in the vicinity together with an indicator to which access point the PDA is connected at a given time and the signal strength for each of them. Each access point was

identified with an x-y-z coordinate and thus the access point id served as the location of the user. The number of data points were not equally distributed over all user, some left their device on all the time, but others only collected one day worth of data. No cleaning of the data was necessary as it was publicly released in a cleaned format. Since some of the users only collected a small amount of data we limited our analysis to 86 out of 275 users. We used the mean number of data points per user as the selection criteria; every user with more data points than the mean is considered in the analysis. It is also to note that 7 out of the 275 users combined 21.4% of the whole data set. We did not consider these 7 users when calculating the mean so as to not exclude more average users. In the end, every user with more than 638.67 data points was considered for analysis. These 86 users combine 85.8% of the whole data set.

Figure 1 shows the analysis of the two most frequent locations averaged over all users. We see that for this data set the two most frequent locations cover 90% of the whole data set. This means that on average the 86 users spend 90% of their time in only 2 locations. This has direct implications on the performance of algorithms, as even the most basic predictor that always predicts the most frequent location for each step would achieve a very high accuracy on this data set.

3.6. Office Occupancy Data Set

This data set was collected as part of a larger sustainability project. We chose a modern, fully instrumented, office building that allows us to remotely retrieve status information. Each room is instrumented with occupancy, temperature, and various heating/cooling related sensors that all communicate through a building automation platform that follows the BACNet industry standard. The information is provided through an Active-X webpage, but also presented in raw html, when accessed using any browser except Internet Explorer. We parsed the webpage through a Python script and extracted the information provided about each room.

For each room we recorded the status of the occupancy sensor in 10-minute intervals over a 4-month period in 2011. A collection of the data at a higher frequency was not possible due to limitations of the BACNet server. We formatted

the data into a data set that records the binary occupancy for each room and 5-minute timeslot by assuming that the occupancy for any two 5-minute timeslots is the same as the corresponding 10-minute timeslot. Our data shows that on average the building was occupied 24.77% of the whole day and 50.98% during business days from 8am to 6pm.

3.7. Evaluation Results

We report on the results for each algorithm individually. The following tables summarize the results by measure, data set, and algorithm. We also report on the algorithm performance if we limit our analysis to weekdays between 8am to 6pm.

Deviating from the original work by Burbey [11], an additional movement label was introduced in our reimplementation of PPM. In Burbey's work non-significant locations were omitted and no data was recorded for that 10-minute time chunk. We believe that introducing this additional label does not change the overall validity of the method, but it gives us the added benefit to predict if a person is moving between two significant locations.

	Accuracy	Quantity	Precision	Recall	Kappa	Baseline
Augsburg – *All Day*	85.55%	75.86%	0.8150	0.8555	0.7334	49.30%
Augsburg – *8am-6pm*	89.36%	82.30%	0.8536	0.8936	0.8119	48.20%
UCSD – *All Day*	97.24%	96.35%	0.9670	0.9724	0.5867	89.70%
UCSD – *8am-6pm*	95.71%	93.81%	0.9455	0.9571	0.5312	83.10%

Table 10 ACE Results by Data Set

	Accuracy	Precision	Recall	Kappa	Baseline
Augsburg – *All Day*	77.36%	0.7792	0.7736	0.4086	77.40%
Augsburg – *8am-6pm*	49.00%	0.5029	0.4941	0.1856	49.06%
UCSD – *All Day*	87.32%	0.8335	0.8732	0.0345	88.52%
UCSD – *8am-6pm*	82.08%	0.7701	0.8208	0.0350	83.77%

Table 11 PPM Results by Data Set

	Accuracy	Precision	Recall	Kappa	Baseline
Augsburg – *All Day*	98.05%	0.4007	0.3473	0.3622	98.30%
Augsburg – *8am-6pm*	94.03%	0.4052	0.3595	0.3497	94.80%
Occupancy – *All Day*	88.50%	0.7975	0.7179	0.6806	78.40%
Occupancy – *8am-6pm*	80.08%	0.7970	0.8176	0.6013	52.80%
UCSD – *All Day*	97.92%	0.7911	0.7694	0.7692	95.20%
UCSD – *8am-6pm*	97.32%	0.7958	0.6735	0.7156	94.63%

Table 12 PreHeat Result by Data Set

3.7.1. ACE Results

The results for the Adaptive Confidence Estimator (Table 1) are calculated on the returned prediction requests, which is why we also report the quantity for this algorithm. It shows a generally high accuracy of between 85.55% and 97.24% across the data sets, high figures that are partly achieved by withholding prediction requests in cases of uncertainty. We can also observe substantial improvements to prediction accuracy over the baseline, especially for the Augsburg data set for which the difference is greater than 40%. If we further investigate the different error cases of false positive and false negative, expressed through precision, recall, and F1-score, we again observe a strong performance of up to 0.9697. This indicates that the ACE algorithm performs well on the portion of the data set for which results are not withheld and is also stable across different data sets. This is also reflected by the Kappa statistic, showing moderate to substantial agreement between the observed and predicted data. Especially Kappa is important for this analysis since it measures the performance of the algorithm on the data set and particularly shows well the algorithm is capable of predicting individual locations independent of the underlying location distribution in the data set. It is to note that the results for the Augsburg dataset show an anomaly for the All-Day results. Our expectation was that predictions during 8AM to 6PM are more difficult since the spatiotemporal structure of the behavior should be more variable and thus harder to predict. The ACE algorithm on the Augsburg dataset made more mistakes outside that time range though which leads to the higher 8AM to 6PM performance. The reason for this can be found in the amount of available data for the prediction:

only 4% of the dataset falls into the nighttime period thus making it more difficult to make accurate predictions.

A caveat to these very promising results is that the algorithm is designed to only returning predictions that it is confident are correct. This is why it is also important to examine the quantity in conjunction with accuracy to gain a full picture of all requests that are not correctly satisfied. For example, if we take the Augsburg dataset and calculate the percentage of data points that lead to either no result or an incorrect result we see that only 64.94% of the All-Day cases return a correct result. As such, a key strength of this algorithm lies in its ability to identify prediction results that have a higher probability of being wrong, which allows it to be effectively used as part of an Active Learning system.

3.7.2. PPM Results

Applying Prediction by Partial Match (PPM) to the different data sets leads to substantial variations in performance – as great as between 49% (Augsburg) and 82.08% (UCSD) in the daytime only data. In addition the difference between the baseline accuracy and the algorithm accuracy is negligible in all cases. Furthermore, even though we observe a fairly high F1-Scores the Kappa values remain low. This is especially apparent for the UCSD data where agreement between the observed and predicted data that is barely above chance. If we compare the results of the *All-Day* and *8am to 6pm* analysis we can see that excluding the nighttime from the prediction analysis substantially lowers the results. For example, for the Augsburg data, the prediction accuracy drops by 28.36%. This accuracy change supports our intuition that including nighttime in the analysis of indoor location prediction algorithms can artificially inflate their performance by considering a non-representative (and non-interesting) scenario involving making predictions about predominantly empty spaces.

We examined the data more closely to account for this performance. The reason for the small difference between baseline and algorithm accuracy is partly attributable to the underlying class distribution. Figure 1 shows the distribution for the two most frequent locations for each data set. In each, the location distribution is skewed towards the two most frequent locations. This is particularly acute in the

UCSD data set as the single most frequent location accounts for over 84% of all entries. This distribution explains the high baseline accuracy in the UCSD data set. The low Kappa value, on the other hand, is due to several qualities of the PPM algorithm. Firstly, the algorithm makes the assumption that an occupant always follows a fixed spatiotemporal routine. More complex routines such as meetings that happen bi-weekly cannot be captured with this approach. Secondly, as PPM's feature space consists only out of time of day (no other temporal, spatial, or spatiotemporal features) or most frequent location in cases of a time of day that is not contained in the training set, it makes numerous wrong predictions for a large number of prediction requests. The prediction performance can be improved by including additional temporal, spatial, and spatiotemporal features.

The performance of PPM on the UCSD dataset also shows us how important it is to include Kappa in the analysis. It is possible to have an algorithm that achieves a high accuracy performance and also high precision and recall values, which would indicate that an algorithm is a good solution for a particular problem. In such a case Kappa can make a difference: is the high accuracy and precision/recall performance achieved by only predicting the majority class or are the class predictions more balanced. A low or high Kappa value is able to express this independent of the underlying class distribution. Measures such as precision and recall are weighted by the class distribution, thus possibly giving a false picture of the algorithm performance. It is also important to not solely rely on the baseline accuracy as a quality measure. An algorithm that achieves close to baseline accuracy, but is capable of predicting minority classes (even in face of a skewed class distribution) might be very valuable depending on the contextual situation.

3.7.3. PreHeat Results

The PreHeat algorithm achieves a very high level of performance on multiple measures on all three data sets. The prediction accuracy is very high and, in most cases, improves over the baseline. However, both the Augsburg and UCSD data sets have an already very high baseline, leaving little space for improvement. Nevertheless the Kappa values indicate PreHeat attains a fair to substantial agreement between the observed and predicted data. Since the Augsburg and

UCSD data set are not ideal for an occupancy based prediction approach due to the high baseline accuracy we collected a novel occupancy based data set from a large university office building. This data set is room-centric and thus independent on the number of current occupants. As we can see PreHeat achieves substantial improvements, of up to 22.72%, over baseline accuracy. Furthermore it yields very high F1-scores and Kappa values, which indicates that the algorithm is not only making a large number of correct predictions, but also producing a very small number of false positives and negatives.

PreHeat's performance was also affected by including nighttime and weekends – non-work hours - in its predictions. If we limit the analysis to weekdays between 8:00 am to 6:00 pm, accuracy drops by up to 8.42%. Figure 2 highlights this point by showing the aggregate accuracy of PreHeat predictions against the time of the day. It shows a clear trend for high accuracy during nighttime with a steep decline during common working hours - the 8:00 am to 6:00 pm time frame.

3.8. Presentation Of Better Practices

As we were able to see in the discussion of the algorithm results each of the different measures highlighted a different aspect of the algorithms performance. Out of this analysis we suggest to use the above measures as a set of better practices for reporting of indoor location prediction results. Many of the points raised also generalize to reporting on performance in other context prediction tasks. As the analysis of the different algorithms and data sets made clear, a single accuracy figure is insufficiently to represent the true worth of a prediction algorithm's outcomes. In fact, the true performance of an algorithm is highly dependent on factors as varied as: the structure the algorithm exploits, the structure that exists within a data set (and the match between these two things), the class distribution within the data set, the types of errors (or non-reports) made by the algorithm, and the performance of the baseline approach to the problem. We believe that key measures that capture these difference aspects can be reported by describing an algorithm's accuracy, the accuracy of the associated baseline prediction, the algorithm's precision, recall and Kappa values. As reported earlier in this paper, most research publications in this area only report the

algorithm's accuracy, which we argue gives a highly incomplete and often optimistic picture of an algorithm's performance. In the following paragraphs we elaborate why we believe that reporting of each of these metrics is important.

3.8.1. Baseline Accuracy

The baseline accuracy provides a lower bar for algorithm performance and enables researchers to evaluate the practical performance improvements an algorithm provides. For example, an algorithm that is improving the accuracy by only one or two percent over the baseline leaves open questions why a more complicated algorithm is needed. On the other hand, situations in which the performance of an algorithm is relatively low, but significantly over the baseline, suggest that the underlying data set offers very little regular structure and prediction systems may struggle to offer reliable results. A discussion of baseline accuracy should also be complemented with analysis about different error cases. For example even an algorithm that does not perform better than the overall baseline can still offer value if it reliably predicts particular critical cases. This also holds true if the baseline accuracy were already very high (greater than 90%), which would indicate a highly structured and predictable dataset. An algorithm that is capable of consistently improving over that high accuracy even by one or two percent can be considered very powerful.

3.8.2. Precision & Recall

Precision and recall represent wide accepted metrics for evaluating the errors an algorithm is making in relation to its correct predictions. Precision expresses the false positive errors, cases for which the algorithm incorrectly predicts a certain class. Recall, on the other hand, captures the false negative errors, cases when the algorithm fails to predict a certain class. Both metrics should be calculated individually for each class and means calculated to provide a global measure of precision and recall for a given prediction task. It is important to note that they are calculated dependent on the underlying class distribution. This means that a skewed dataset (a dataset that has one very dominant class) also will have high precision and recall if they are high for the dominant class. Even given this restriction both metrics allow evaluation of the errors an algorithm is making. This

is particularly important if the errors are not uniformly important in the algorithm use scenario. The application space of an algorithm defines the importance of each error class.

Lets assume for example an indoor location prediction algorithm is used for predictive automatic room temperature control in a large office building. Recent years have shown that companies heavily focus on employee satisfaction, emphasized by employee perks such as free food or fitness and leisure facilities that are situated in corporate campus buildings. The thermal comfort of an individual is a large impact factor on an employee's happiness and satisfaction as the success of products such as Comfy[1] has shown us. An algorithm that predicts when a person returns to their office and that enacts changes based on this prediction can make to possible errors: it predicts that the occupant returns to their office even though they are not (False Positive) or it predicts that the occupant does not return to their office even though they are (False Negative). Each of these two errors has a different impact on the overall system. The former results in larger energy expenditure, because the office temperature is being adjusted without having a person present. The latter might save energy, but has potential negative impact on a person's thermal comfort and satisfaction. Under the scenario of increased user satisfaction a system developer would choose an algorithm that has a very low False Positive rate (and subsequently high precision) even if it means this performance is achieved with a higher False Negative rate (meaning lower recall). As long as the resulting control decisions still have a positive impact on the building's energy consumption, compared to a schedule-based control, the system will be a success, because it maintains the occupant's thermal comfort and saves energy.

As we can see with the above example, using precision and recall to characterize errors in this way allows for a more fine-grained algorithm performance analysis. This will allow researchers to identify gaps in the current state-of-the-art as well as practitioners to choose the right algorithm for the right situation. Reporting on

[1] https://gocomfy.com

accuracy alone as we observed in previous literature has a negative impact on pushing the state of the art and forces system developers to test a wide range of algorithms for their suitability for a particular problem.

3.8.3. Kappa Statistic

While the precision and recall evaluate algorithm performance in general dependent on the class distribution, the Kappa statistic allows evaluation of the performance of an algorithm on a specific data set (i.e., class distribution independent). Basically, Kappa expresses how an algorithm performs independent of the underlying class distribution in the data set the algorithm is tested on. This is especially important if the underlying distribution is biased towards one class and the algorithm is not able to predict any of the minority classes. For situations like this we will need a metric to express how much the observed data agrees with the predicted data. In other words, Kappa measures how well the algorithm predicts not only the majority class, but also the minority classes. The results of our analysis of the PPM algorithm on the UCSD dataset highlighted the importance of calculating this metric. Even though the algorithm has a high accuracy as well as high precision and recall, the low Kappa indicates that agreement between the observed and predicted data is close to chance. The reason for this is that the algorithm performs poorly on the relatively infrequent minority classes. In such situations, the relative importance of predicting the majority and minority cases need be weighed qualitatively before determining the suitability of the algorithm.

A high Kappa value on the other hand signifies a high agreement between the observed and predicted data and can be used as an indicator for a high performing algorithm. An algorithm that has an observed Kappa value in the mid ranges (0.4 to 0.7) would require us to look at the additional metrics mentioned above. But again the evaluation has to be conceptualized by the underlying problem space. For example if Kappa is very low, but we observe high baseline accuracy, accuracy, and precision and recall, we can surmise that the algorithm is primarily predicting the majority class and that the underlying dataset is skewed towards that class. If the majority class is the most relevant class the algorithm might be enough for the problem space.

3.8.4. Separate Analysis Into Day-Time and Night-Time

As we could see in both in our analysis and the analysis performed by Oh *et al.* [52] the performance of an indoor location algorithm changes depending on the time of day. The location of a person during nighttime is much more predictable since there is a lower probability for the user to be in their office at night. This results in a higher accuracy for these times, which potentially biases the overall performance if it is not factored out during the analysis stage. The effect of this is especially observable for the PreHeat results on the occupancy dataset. The accuracy drops off by over 15% between nighttime and daytime. If the results are limited to daytime hours the accuracy is over 8% lower during daytime hours compared to the whole day performance. These examples show that it is important to limit the analysis to the appropriate analysis frame. If the dataset describes an office environment, which follows certain occupancy patterns, it is important to restrict the analysis to these hours or present a comparison between all-day vs. daytime results.

3.8.5. Conclusion

Prior state of the art indoor location prediction algorithms achieve relatively high performance across multiple metrics. Nevertheless the results highlight the importance of diverse metrics since a one-dimensional analysis can hide potential algorithmic short-coming or overreliance on an existing structure in the datasets. From all the algorithms that include both location and time in their prediction, PreHeat exhibits the highest performance, which is why we also use that algorithm as our point of comparison later on. As stated before PreHeat exploits the underlying spatiotemporal structure of human behavior to make predictions about future events. To further investigate this approach the next chapter presents an indoor location prediction algorithm, Indoor-ALPS, that utilizes various spatiotemporal features for its predictions.

4. Modeling of Spatiotemporal Behavior

The Importance of Spatiotemporal Features: Indoor-ALPS

Predicting occupant indoor locations and when occupants will transition between these locations is challenging. Existing systems (see Chapter 3) that monolithically approach this task, combining a prediction of where a user will go with one of when they will go there, perform poorly with prediction results that are close to the performance of a majority predictor (*i.e.*, predicting that the occupant will always stay in his most frequented location). This is likely because occupants' routines within the interior of a building, such as a workspace, can be very complex. They can change substantially from weekday to weekday, and encompass numerous destinations and transitions in relatively short spans of time.

To simplify this problem, most prior work has focused on predicting an occupant's next location without taking into account when these transitions will take place [44,52,57,71,73]. Unfortunately solving this problem alone precludes many applications that rely on knowledge about transition timings. For example, a system that proactively heats a room in advance of an occupant's arrival [20] would not know whether the person was coming in five minutes or two hours, and could result in either significant discomfort (if heated too late), or significant wasted energy (if heated too early).

As mentioned before one factor that makes it difficult to accurately predict an occupant's transition time into a space is the fact that indoor routes are typically short (meters to hundreds of meters) and traversed rapidly (in seconds to minutes). This is in contrast to the problem of outdoor location prediction (*e.g.*, [9,24]), for which travel times and distances are long enough that a useful prediction algorithm only needs to predict the next significant location after the user has already departed. This allows the use of features such as the currently traversed path to

make predictions, which are mostly not useful to applications of indoor prediction, due to the shorter indoor transition times. This also means that many of the analytic techniques that perform well outdoors are inappropriate, or simply do not work well, when applied to indoor scenarios. Making exact predictions about when a person transitions is a hard problem due to the variability in length of stay at a location. How long a person stays at a given location is dependent on their spatiotemporal routine and can vary greatly from weekday to weekday and fluctuates depending on prior events that occur during a day. For that reason it is important to consider a wide range of spatiotemporal features and not only the current location and time.

To explore the theory that the use of a range of spatiotemporal features improves the performance of indoor location prediction algorithms, we created a novel spatiotemporal indoor location prediction algorithm called Indoor Adaptive Location Prediction System (*Indoor-ALPS*) [37]. Indoor-ALPS tackles the challenge of accurate temporal and spatial prediction by splitting the problem into two separate steps: 1) For a given time interval, Indoor-ALPS predicts whether an occupant will stay in her space for at least that long and 2) for a given location, Indoor-ALPS predicts the next location she will transition to. The algorithm then combines these two independent predictions. It first predicts whether or not an occupant will stay at a location for at least a given time window and then predicts her destination. Unlike previous approaches that perform these two steps simultaneously, Indoor-ALPS's decomposition of the problem allows it to learn which contextual features are the best predictors for the individual temporal and spatial problems. We argue that dividing the problem in this way will lead to improved prediction performance.

To verify this claim, we compared Indoor-ALPS to Prediction by Partial Match (PPM), a state of the art indoor location prediction algorithm [11]. We evaluated my approach using a range of time windows from 10 to 90 minutes, in an effort to show its applicability for different applications. My analysis showed that Indoor-ALPS improved the overall prediction accuracy by 6.2% over PPM. Indoor-ALPS was particularly strong when considering temporal look-aheads of 10 to 30 minutes, when it led to a significant mean accuracy improvement of 10.7% with a maximum

improvement of 12.9%. In the following subsections we will describe the algorithm and present the results of the comparative analysis between Indoor-ALPS and PPM.

4.1. Algorithm Description

As mentioned in the introduction, Indoor-ALPS splits the prediction problem into two independent steps, which it later combines. First, for a given time interval, Indoor-ALPS predicts whether the user will *stay for at least that long* in the current location. Second, for a given location, it predicts where the user will go next. Finally, Indoor-ALPS combines these predictions.

More specifically, to make both the spatial and temporal predictions for each day, we use the same algorithm (independently for each type of prediction). It uses a combination of two approaches that can help improve classification accuracy: *ensemble prediction* (training and combining results from several classifiers); and *incremental learning* [27] (using each newly recorded data point as part of the training dataset after the prediction for that data point is completed). The basic algorithm is as follows:

1. Take all the data in the dataset up until the current day (for which prediction is being performed), and split it into two non-overlapping data sets: optimization and training.

2. Using this data, identify the best feature-subset for each classifier used in the ensemble algorithm: Decision Tree, 3-Nearest Neighbor, Support Vector Machine, and Gradient Boost.

3. Train each of the four classifiers, using their feature set tailored to them in Step 2.

4. Using each classifier, make predictions for the current day.

5. Calculate the ensemble prediction [43] using the results of each individual classifier.

6. Repeat from step 1 until all days in validation set are predicted.

We use a portion of the data for each user to create initial models and validate on the remaining data for that user. Initially 10 days of data are used and equally split into the optimization and training data set. After feature selection using the optimization data set, the models for each algorithm are created using the training set and predictions are performed on the validation day. After the predictions are complete for that day, that day's data is added to either the optimization or training data sets. These days of data are added in an alternating fashion, first day to the training set, next day to the optimization set, and so on, to maintain an almost-even split of the data between optimization and training.

4.1.1. Feature Selection

Predicting if a person stays at a location for a given amount of time and where they transition to if they do not stay is highly dependent on their temporal-spatial routine. Making a temporal or spatial prediction depends on factors such as day of week, current location, or even the arrival time at the office in the morning. Indoor-ALPS uses ten temporal-spatial features that capture these influences:

- Current location (*L*)
- Time of arrival at *L*
- Minutes passed since arriving at *L*
- Current time
- Current day of the week
- Arrival time in the building for the current day
- The number of significant locations the occupant visited previously for the current day
- Previous two significant locations
- Stay duration at the previous significant location
- Length of the transition time to the current location

All times are expressed as discrete ten-minute timeslots starting from midnight and significant locations are defined as locations in which the occupant frequently stays for at least ten minutes [4].

Since we do not know if all features are equally important for each user and situation, our algorithm uses Sequential Floating Forward Selection (SFFS) [58] to find the most relevant features from the set of ten features. SFFS is a greedy algorithm that adds one feature per iteration to the already selected feature subset. After each new feature has been selected SFFS checks whether a subset of already selected features can be removed without decreasing the performance. We used an objective function that maximizes accuracy. We allowed SFFS to create a feature set consisting of between one and ten features based on this objective function. Our algorithm applied SFFS independently for four different machine-learning algorithms: Decision Tree, 3-Nearest Neighbor, Support Vector Machine, and Gradient Boost.

Our algorithm re-evaluates, after each predicted day what is the best feature subset for each algorithm. By applying this incremental learning approach, the algorithm has the opportunity to react to changes in users' temporal-spatial routines.

The location data used for the prediction is formatted into discrete 10-minute time slots, both to make it easier to compare our algorithm against previous techniques [11] and to keep in line with the definition for a significant location [4]. For each stay at a location we interpolated data points that represent the current stay duration at a location. For example, let us assume that the user arrived at *Location A* at *12:50pm* and transitioned to *Location B at 04:10pm*. From this recorded data we interpolate the data as follows:

- 12:50pm Location A; Duration 0 minutes
- 01:00pm Location A; Duration 10 minutes
- …
- 04:00pm Location A; Duration 190 minutes

- 04:10pm Location B; Duration 0 minutes

4.1.2. Transition Time Prediction

In order to predict when a person will transition from one location to the next, our algorithm answers the question “Will the occupant stay at the current location for the next *n* minutes?” This is realized as a binary prediction with 1 representing *yes* and 0 representing *no*. Recall that the algorithm uses an ensemble method. For this prediction, this consists of a simple mean of the predictions of the four classifiers. If the mean prediction returned by the ensemble method >= 0.5, Indoor-ALPS assigns 1 as a prediction result, and it assigns 0 otherwise.

However, as different applications are likely to require predictions about different stay durations, we trained algorithms and performed predictions for nine specific time windows: n = 10, 20, 30, …, 90. This approach can be used to predict the time for transitioning from the current location. We predict stay duration for successively larger times, and for the time *t* when the ensemble classifier returns a 0, we infer that the user will leave only after that time.

4.1.3. Prediction of Next Location

The prediction of the next significant location works similarly to the prediction of the transition time. Again using the algorithm described above we ask the question “What is the next significant location the user will transition to?”. Unlike the binary prediction of staying in the current location for a particular time window, here we are solving a multiclass problem, where the classes are the set of significant locations. We use the same interpolated data we created for the temporal prediction and record the next location for each data point. For example in the data snippet above, each interpolated data point has Location A as the next location.

For this prediction, our ensemble classifier uses a majority voting approach. Each algorithm makes a prediction and the most common prediction is used, rather than the average as was used with the transition time prediction. If there is no clear majority, we choose the location at random from among the predicted locations.

4.1.4. Temporal-Spatial Prediction

To use the two independent predictions for temporal and spatial prediction we chain them together as follows: first we run the temporal prediction to determine if the occupant will stay at the current location or not, and secondly, if the occupant is predicted to leave, we run the spatial prediction to determine where she will transition to. We now describe the results from applying our algorithm to a real-world dataset.

4.2. Indoor-ALPS Evaluation

In order to evaluate and better understand the performance of Indoor-ALPS, we compared it to Prediction by Partial Match (PPM) [11], a state of the art algorithm for predicting occupancy and location transition. We selected a publicly available data set, the Augsburg Indoor Location Tracking Benchmark [57] to compare the algorithms.

4.2.1. Using Prediction by Partial Match as a Benchmark

Prediction by Partial Match (PPM) [11], to our knowledge, has the highest reported accuracy for the task of predicting when an occupant transitions from one location to the next and where the occupant transitions to. PPM is based on a 1st order Markov Model. The input data for this algorithm is formatted into 10-minute time slots and for each time slot PPM calculates the frequency that each location was recorded for that time slot. In order to make a prediction, the algorithm takes a time slot as an input and returns the location with the highest frequency.

We used PPM to make both temporal and spatial predictions. As we explained previously, PPM takes a discrete timestamp and predicts the most frequent location for that timestamp. To make *temporal* predictions, given a current time index ***t*** and look-ahead window ***n*** (to answer the question: “Are you staying at this location for the next ***n*** minutes?”), we query PPM to predict the location for the next ***t+i*** (i=1,..,***n***) time slots. If the resulting location for any of these ***n*** predictions is different from the current one, we assign 0 as the result (person is not staying for the next ***n*** minutes) or 1 otherwise. In order to make *spatial* predictions given a current time index ***t***, we query PPM to predict the location for the next ***t+i***

(i=1,..,143, corresponding to a one-day look-ahead) time slots. The first predicted location that is different from the current one is returned as the next location.

4.2.2. Evaluation Data Set

The Augsburg data set has been used extensively in previous attempts to predict location [52,55,61,73]. We chose it for our analysis as it enables easier comparison of our algorithm to past and future efforts to address indoor location prediction. The data set was collected using a smart doorplate concept: each user was tracked using an RFID card upon entering a space (*e.g.*, office, corridor, kitchen). Four university office workers who worked on the same floor were tracked for an average of 6.75 weeks (SD=1.71). Due to the nature of the data collection, this data set already contains distinct locations, thus preprocessing is not needed to extract them. However, as we care about significant locations [4], we performed a preprocessing step to filter out data where the occupant left a room for less then 10 minutes and returned to the same room.

4.3. Results

We split the analysis of our proposed approach into three parts: temporal, spatial, and combined temporal-spatial. We performed a statistical analysis on the accuracy of each algorithm. We then report other measures to support the discussion about the advantages and limitations of each individual algorithm.

In all of our statistical analysis we used One-way and Two-way Repeated Measures ANOVAs, as appropriate. We used Repeated Measures ANOVAs because algorithms were trained on data for each individual user, and the training and testing data split for both algorithms was the same. We ensured the normality of data using the Shapiro-Wilk test. We ran Mauchley's test for sphericity and performed Greenhouse-Geisser correction when the sphericity assumption was violated. All *post-hoc* pair-wise comparisons were done using paired t-tests with Bonfferoni correction. We report only main effects and p-values in the text for succinctness.

4.3.1. Temporal Analysis

Different applications require different temporal look-ahead windows. For example, the look-ahead window for indoor temperature control is based on the average

time it takes to raise or lower the temperature in a room to a comfortable temperature, and this can vary based on the current time of day, outside temperature, and position of the room in the building (is the room inside of the building, does it have windows, *etc.*). To show how Indoor-ALPS performs under different temporal thresholds we report the results for nine different look-ahead windows, from a 10-minute to a 90-minute look-ahead. We compare our results against PPM and a 0-R predictor (majority class predictor).

The ground truth for the temporal prediction is tied to the look-ahead window and consists of a series of 0's and 1's. Given a look-ahead window *n* and a data point, the ground truth for that data point is 1 if the person stays at the current location for the next *n* minutes. If, on the other hand, the remaining stay-duration is smaller than *n*, we assign 0 for the ground truth.

We expect that the 0-R predictor has very high accuracy for lower look-ahead windows, since the only instances when an occupant is predicted to leave a location is shortly before the transition time. This makes the prediction task especially difficult for lower thresholds, because there is little data available for the *leave*-class.

Figure 2 shows the average accuracy across all four users for all 9 look-ahead windows by algorithm. Our tests found a main effect of the *Algorithm* on the *Accuracy* ($F_{(1,3.01)}$=11.56, p=.0422, η_p^2=.79); Indoor-ALPS overall mean accuracy of 88.2% was significantly higher than PPM (mean=83.6%, p<.0001) and 0-R (mean=83.3%, p<.0001). Our tests did not find a significant difference between PPM and 0-R (p>.9999).

Our tests also found a main effect of *Look-Ahead* on *Accuracy* ($F_{(1.13,3.39)}$=144.07, p=.0006, η_p^2=.97). As the look-ahead increased, the accuracy of all three algorithms decreased. This makes sense, as it is harder to predict mobility further into the future. Our tests also found simple effects of *Algorithm* on *Accuracy* for all individual *Look-Aheads*. In particular, Indoor-ALPS was significantly more accurate than 0-R and PPM at the 10 minute, 20 minute, 30 minute and 70 minute look-aheads (See Figure 2, p<=0025 in all cases).

Thus, Indoor-ALPS performs better overall than PPM and 0-R, and is especially better for small look-ahead windows of 10 to 30 minutes. The average accuracy difference between Indoor-ALPS and PPM for the first 3 look-ahead windows is 9.2% (SD=2.3%) and 3.9% (SD=0.2%) for the higher look-ahead windows.

Figure 3 shows Precision and Recall for all look-ahead windows for Indoor-ALPS as well as PPM. Note by definition, 0-R will always have a Recall of 0. We see a slight improvement for the Precision of Indoor-ALPS over PPM for the first five look-ahead windows after which PPM has a slightly higher Precision. For the smaller look-aheads, PPM makes more false positive errors, *i.e.*, predicting that an occupant is staying when they are not. For the larger look-aheads, Indoor-ALPS makes more false positive errors. However, Indoor-ALPS has a consistently higher Recall than PPM with an average Recall across all look-aheads of 96.6% (SD=2.1%) for Indoor-ALPS and 83.6% (SD=2.9%) for PPM. PPM's lower Recall means that it makes more false negative errors, predicting that an occupant is leaving a location a lot earlier than they actually do.

Figure 4 shows the Kappa results by look-ahead window and algorithm. As expected from the previous Accuracy, Precision and Recall results, Indoor-ALPS has higher Kappa values for the lower look-ahead windows and slightly worse Kappa values for the higher look-ahead windows. The Kappa values for PPM increase as the look-ahead gets larger. It is also of note that the standard deviation for the Kappa values is fairly uniform for PPM (avg.=0.075, SD=0.013), while the standard deviation increases for higher look-ahead windows for Indoor-ALPS (avg.=0.087, SD=0.052). In the discussion section, we will provide a rationale for why this occurred.

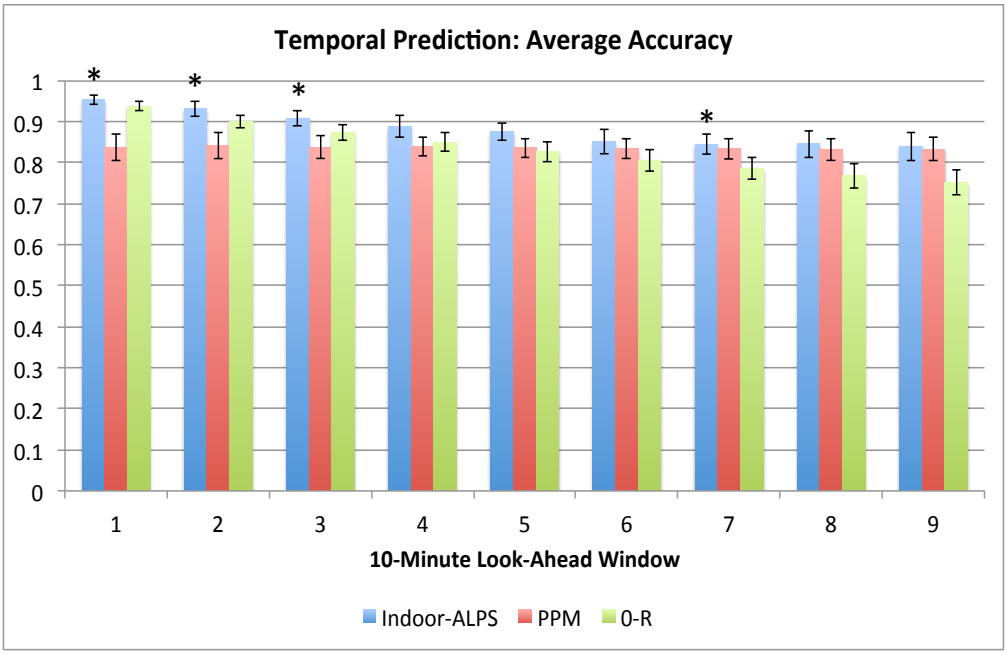

Figure 2 Average Accuracy by Look-Ahead Window for Temporal Prediction. Starred bars indicate statistically higher values (p<=.025).

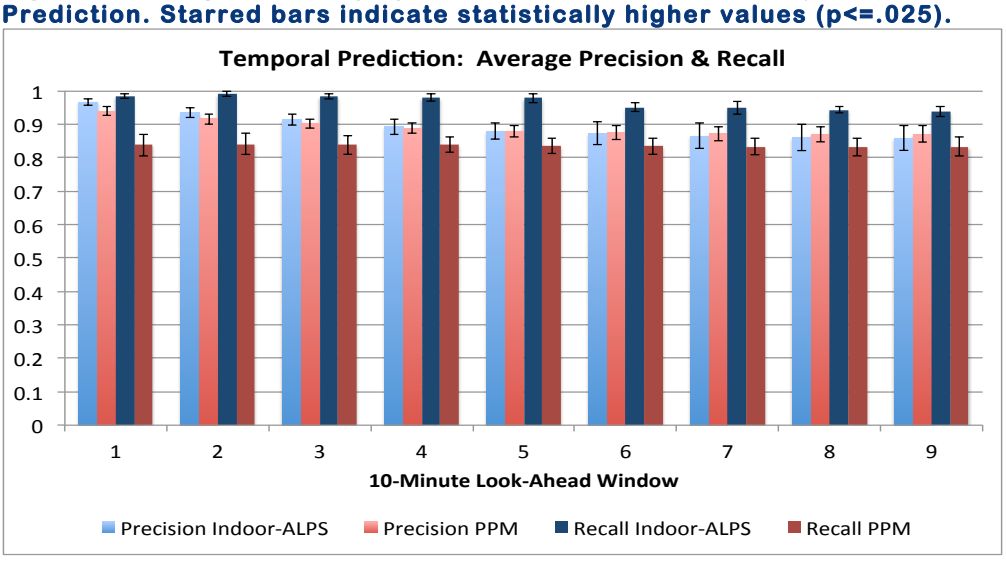

Figure 3 Average Precision & Recall for Temporal Prediction

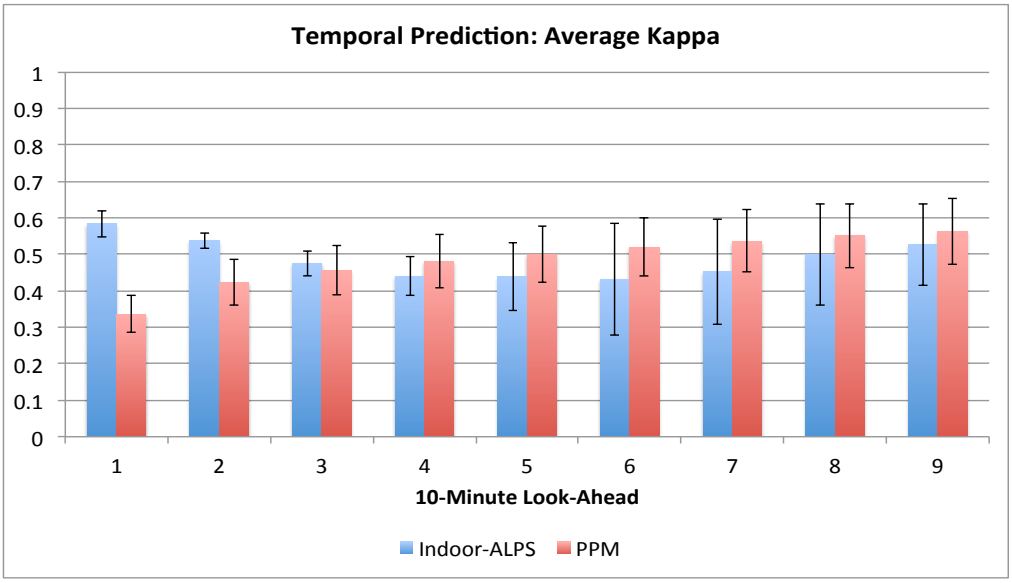

Figure 4 Average Kappa for Temporal Prediction

4.3.2. Spatial Analysis

In order to evaluate the performance of the spatial or next significant location prediction, we again report on the Accuracy, Precision, Recall, and Kappa statistic. The ground truth for this analysis is the next significant location that follows the current location. Figure 5 shows the accuracy by user for each algorithm. As we can see, both Indoor-ALPS as well as PPM outperform 0-R. The accuracies for Indoor-ALPS and PPM are quite similar, with PPM outperforming Indoor-ALPS for User 3. Note that we cannot run statistical tests on accuracy across our users for our spatial analysis as we only have 4 users and thus only 4 values per algorithm. We also analyzed Precision and Recall (see Figure 6) and found a similar picture as with the accuracy. Looking at Kappa (see Figure 7) the algorithms again have similar performance. We can see that both Indoor-ALPS and PPM have a substantial agreement between the observed and predicted data for users 1 and 2, a moderate agreement for user 3, and Indoor-ALPS has a slight agreement and PPM has a fair agreement for user 4. Overall, we can draw the conclusion that PPM offers modest improvements over Indoor-ALPS for users 3 and 4, and performs similarly for users 1 and 2.

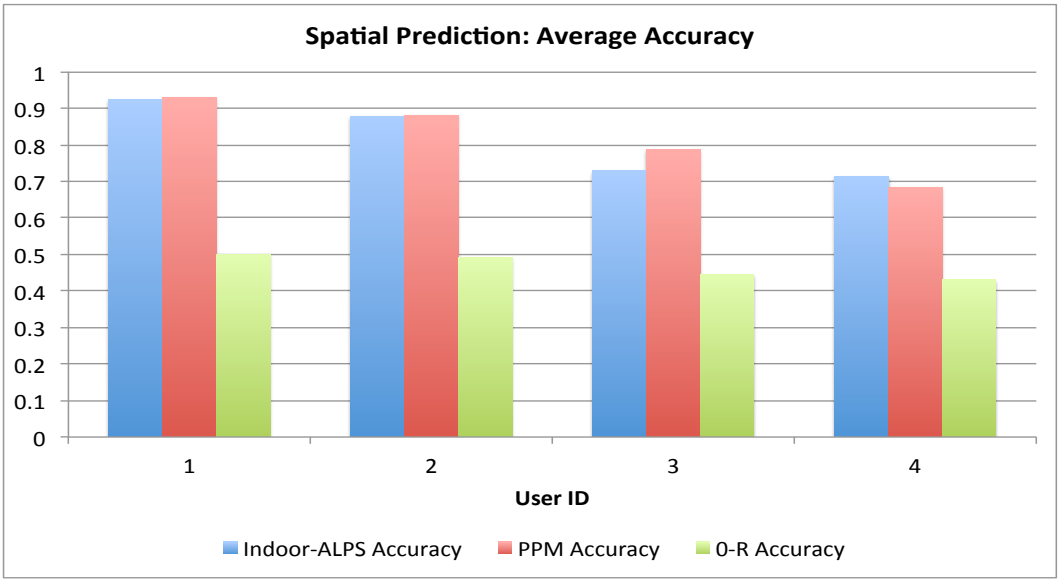

Figure 5 Accuracy by User ID for Spatial Prediction

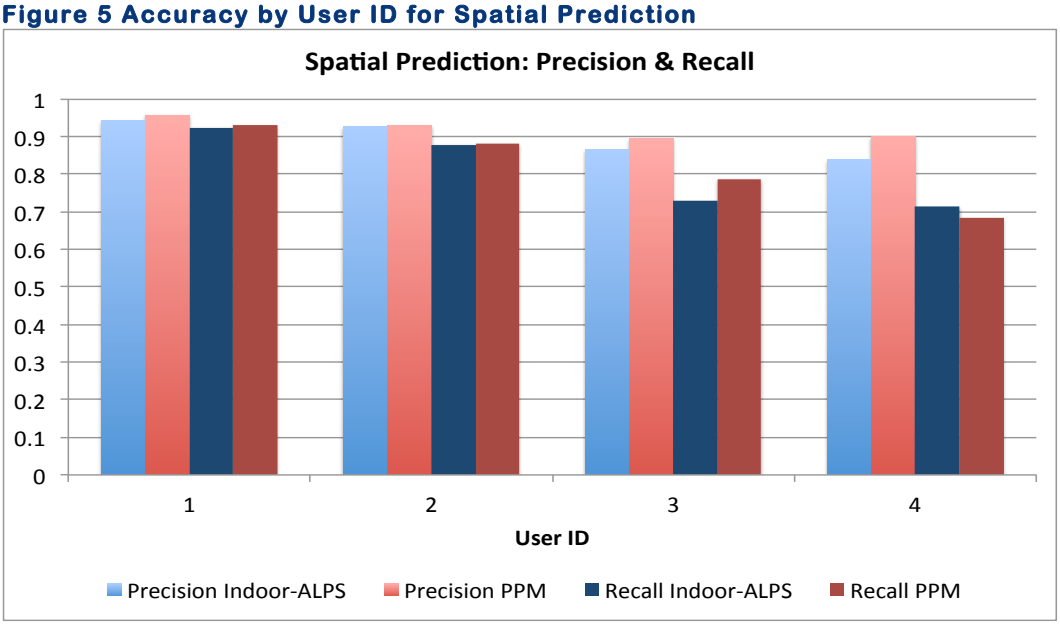

Figure 6 Precision & Recall by User ID for Spatial Prediction

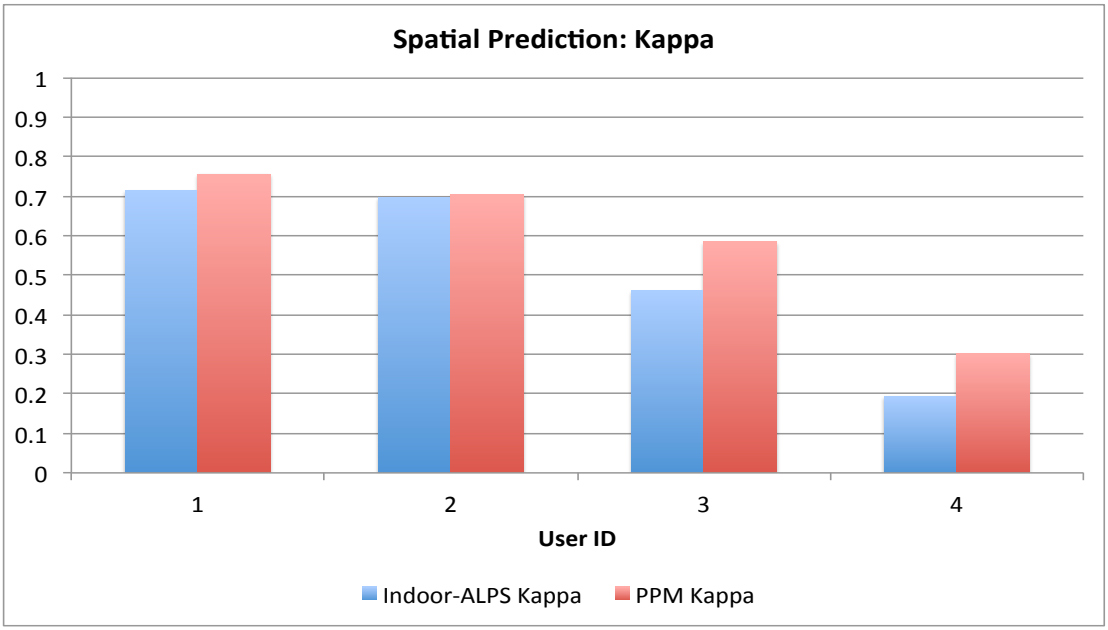

Figure 7 Kappa by User ID for Spatial Prediction

4.3.3. Temporal-Spatial Analysis

Our algorithm combines the results of the temporal and spatial prediction to predict if an occupant will stay at a location for a given time duration, and if the occupant is predicted to leave, to which location the occupant is transitioning. To evaluate the algorithms, we first calculated how often they correctly predict that an occupant is staying, and when they correctly predict that the occupant is leaving, how often they correctly predict the next location. Figure 8 shows the average prediction accuracy by look-ahead window. Note we did not include 0-R in this analysis as the results are identical to the temporal 0-R, since 0-R predicts that the user is always staying and thus no location prediction is performed.

Our tests found a main effect of *Algorithm* on *Accuracy* ($F_{(1,3)}$=65.65, p=.0038, η_p^2=.96), indicating that Indoor-ALPS (mean=85.3%) overall performs better than PPM (mean=79.1%). Our tests also found a main effect of *Look-Ahead* on the *Accuracy* ($F_{(8,24)}$=66.75, p<.0001, η_p^2=.96), but the *post-hoc* pairwise comparison did not find any significant difference between *Look-Aheads*.

The analysis of the algorithms and look-ahead windows also showed a significant interaction between *Algorithm* and *Look-Ahead* ($F_{(1.75,5.26)}$=15.58, p=.0067, η_p^2=.84). Again, we only further compare algorithms across different look-ahead windows.

Indoor-ALPS was significantly more accurate than PPM for look-ahead windows of 10 to 80 minutes (all $p<.05$) (Figure 7). Our tests only failed to find a significant difference for the look-ahead window of 90 minutes ($p=.63$). The greatest difference can be observed over the first three windows with an average difference of 10.7% (SD=2.2%), with a maximum improvement of 12.9%. For the first five windows, it is 9.1% (SD=2.7%). This means that combining temporal and spatial prediction in Indoor-ALPS is more accurate than PPM overall, and although the performance difference gets smaller as the look-ahead grows, Indoor-ALPS performs better than PPM in almost all look-ahead windows.

Figure 9 shows the Precision and Recall results for each algorithm. We can see that Indoor-ALPS is slightly worse on Precision with an average of 0.852 (SD=0.051) compared to 0.864 (SD=0.049) for PPM, but it is much better for Recall with an average of 0.915 (SD=0.039) for Indoor-ALPS and 0.794 (SD=0.010) for PPM. Indoor-ALPS' Recall is on average 0.121 (SD=0.032) greater than PPM's Recall, which only slightly changes from one look-ahead window to the next. The lower Recall indicates that PPM leads to more false negatives.

Compared to PPM, Indoor-ALPS has a higher Kappa (see Figure 10) for the lower look-ahead windows and a slightly lower Kappa for the higher look-ahead windows. Similar to the temporal prediction, the standard deviation for Indoor-ALPS' Kappa increases as the look-ahead grows, while the standard deviation for PPM's Kappa is fairly stable with an average of 0.051 (SD=0.006).

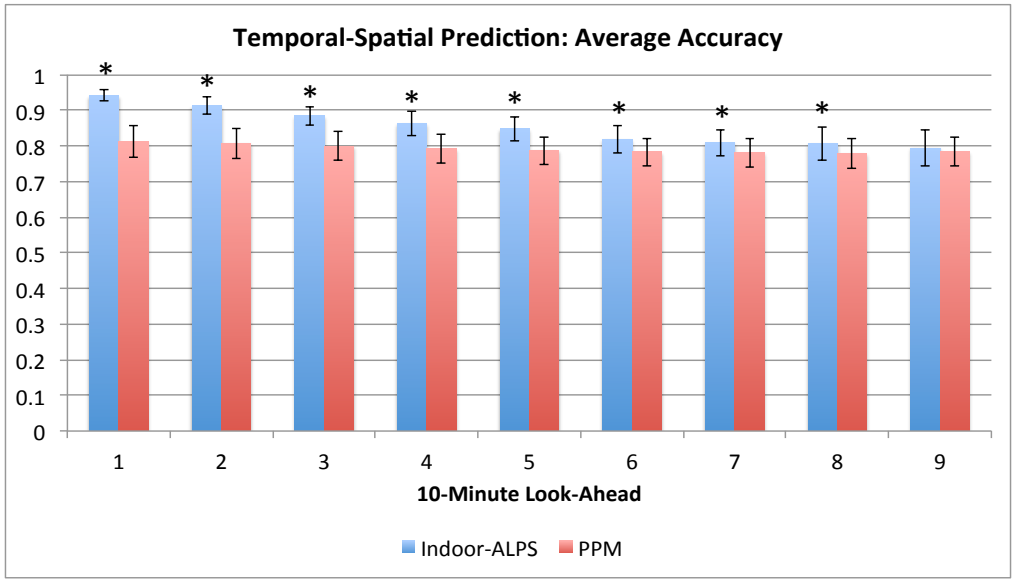

Figure 8 Average Accuracy by Look-Ahead for Temporal-Spatial Prediction. Star indicates statistically significant difference (p<.05) .

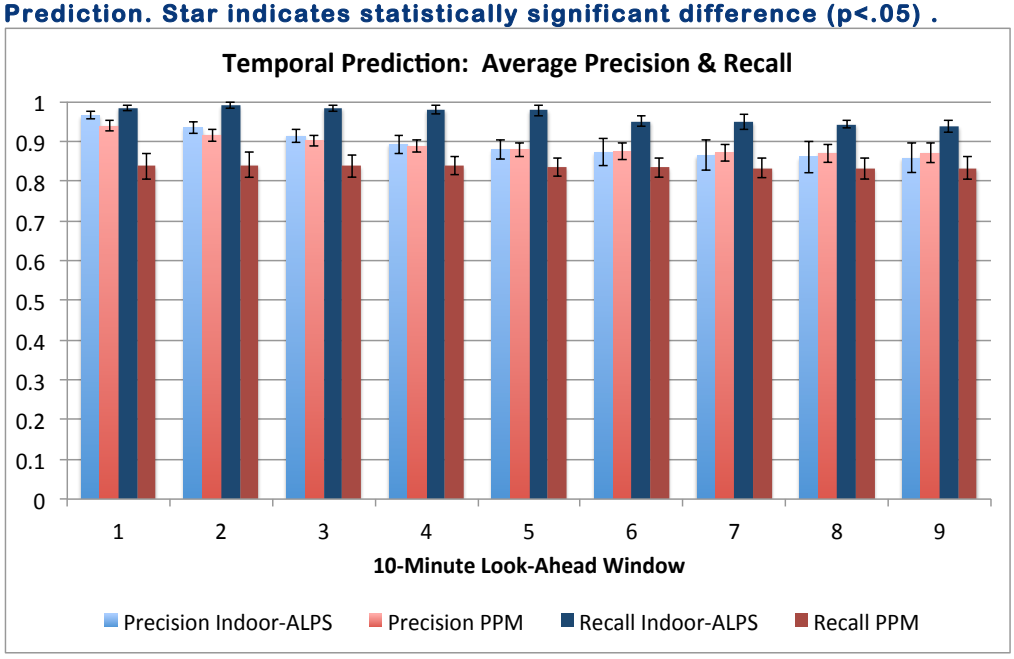

Figure 9 Average Precision & Recall by Look-Ahead Window for Temporal-Spatial Prediction.

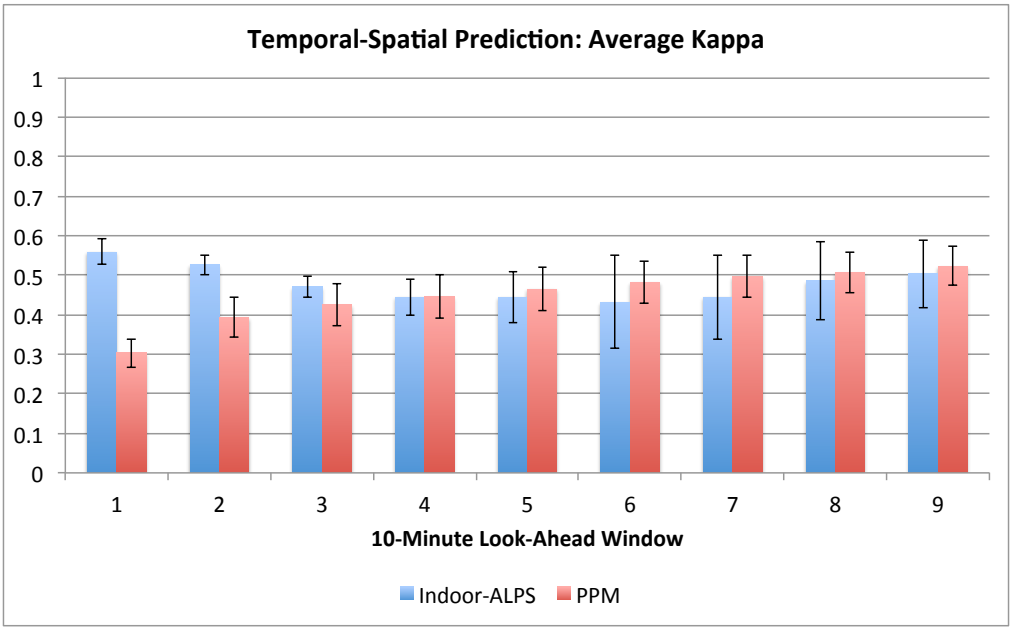

Figure 10 Average Kappa by Look-Ahead Window for Temporal-Spatial Prediction.

4.4. Discussion

Our performance analysis of Indoor-ALPS compared to PPM shows that Indoor-ALPS achieves higher overall temporal and temporal-spatial prediction performance. Furthermore, Indoor-ALPS performs particularly well for the look-ahead time windows of 10 to 30 minutes. Especially for our main prediction goal, temporal-spatial, it outperforms PPM for all window sizes up to 80 minutes in terms of accuracy. This discussion will highlight the advantages and disadvantages of Indoor-ALPS and explain why PPM performs better in some cases. We also illustrate how Indoor-ALPS can be used in practice in an application.

4.4.1. Temporal Prediction

As we described in the results section, Indoor-ALPS achieves a very high performance increase over PPM on multiple measures for the temporal prediction. This is especially true for lower look-ahead windows for which Indoor-ALPS had a mean accuracy gain of 7.3% (SD=3.1%) over PPM. Furthermore our algorithm also improves over 0-R even for lower look-ahead windows for which the 0-R accuracy is already very high.

However, the Kappa results for higher look-ahead windows were worse than PPM. As highlighted earlier, we observed a higher standard deviation for the higher look-

aheads for Indoor-ALPS. This indicates that there were significant differences in the performance of our algorithm for one or more users. Careful analysis of the results shows that user 3 was responsible for these deviations. In fact, if we remove this user from the analysis and recalculate the average Kappa for both Indoor-ALPS and PPM for look-ahead windows of 50 to 90 minutes, we see that Kappa increases by 0.057 for Indoor-ALPS while it only increases by 0.015 for PPM when compared to the results for all four users. An analysis of user 3's data revealed that in the validation data, the user frequently went to two additional new locations, which were not present in the initial training data. If we analyze the selected features of Indoor-ALPS for each validation day and look-ahead window we notice that the current location is one of the most frequent features picked by the feature selection algorithm. Since the data for these new locations is relatively sparse in the training data, our algorithm frequently predicts that the occupant stays. Even with incremental learning, it takes a while to collect enough data about the new location to make accurate predictions. This is also reflected in the Precision, which is lower than the average precision of users 1, 2, and 4 by 0.055.

PPM on the other hand is agnostic to the current location when it makes a prediction for a given time slot. It only uses discrete time to make a prediction. As long as the predicted location differs from the current one it will predict that the occupant leaves within a given look-ahead window. This is why PPM is more robust to changes in a person's routine. For our future work, we plan to extend Indoor-ALPS and allow it to react to changes in the user's routine by falling back to a frequency-based model.

Even without including user 3 in the results we still see that Kappa is slightly higher for PPM on look-aheads of 40 to 70 minutes (average difference of 0.042). This behavior can be explained by looking at the Recall for PPM. We see that PPM has a much lower Recall, indicating that the algorithm very frequently predicts that the user will leave her location even if she actually stays (False Negative errors). By doing so, it correctly predicts more of the Leave-class, which is the minority class in the data set, but at the expense of the Stay-class, resulting in a slightly higher Kappa. The slightly lower Precision shows that Indoor-ALPS makes the opposite

errors; when the data shows that the occupant leaves within the next *n* minutes, our algorithm sometimes incorrectly predicts that the occupant will stay.

4.4.2. Spatial Prediction

The evaluation of the spatial prediction showed that Indoor-ALPS and PPM outperform 0-R and both achieve comparable performance. Only for user 3 we see that PPM achieves a higher accuracy than Indoor-ALPS, with a difference of 5.9% between the two. The cause for this is the same as was described for the temporal prediction for user 3. Since the prediction algorithm uses the current location as a feature, it has difficulties with two new locations. The locations that are transitioned to when leaving these two new locations are very frequented (or majority) locations for this user, which is why PPM is better able to handle the new situation since it predicts the majority location. One potential way to improve the next location prediction is by leveraging the temporal prediction along with Active Learning. In situations when the temporal prediction predicts that the occupant is leaving the current location and the prediction is uncertain about the most likely next location, it can ask the user for input to improve its ability to learn.

4.4.3. Temporal-Spatial Prediction

Analyzing the overall performance of the temporal-spatial prediction, the task we set out to solve with Indoor-ALPS, we can see that our algorithm is significantly more accurate than PPM, where it performs much better particularly on the lower look-ahead windows. Our proposed algorithm addresses the following problem: predicting if an occupant will stay at their current location for a specified time frame and, if they are predicted to leave, predicting also the indoor location they will transition to within that time frame. Our algorithm with its high prediction performance can support a large class of compelling applications, including proactive traffic notifications, proactive heating, automated receptionists, and *ad-hoc* meeting support. We now look in detail at how Indoor-ALPS supports proactive heating.

Using Indoor-ALPS for Proactive Heating

The importance of the different look-ahead windows is dependent on the application context. For example, efficient temperature control in domestic

environments requires a look-ahead of, on average, 60 minutes [38] to change temperature by 10°F. In office environments with zoned temperature control, the heat-up time is usually smaller due to the smaller volume of the space that is heated and secondary heating effects from adjacent rooms. A look-ahead window of 30 minutes is already enough to positively affect a building's overall energy consumption while minimizing the impact on the occupant's thermal comfort. Our algorithm is particularly suited to solve this problem.

Let us assume the HVAC system in a building needs 30 minutes to change the temperature by 10°F, which requires a 30 minute look-ahead window for knowing if a person will transition to a new space. Given a particular location, our algorithm would make temporal predictions every 10 minutes in order to determine if the person is staying at their current location for at least 30 minutes. As soon as the prediction result changes from yes to no in consecutive queries, Indoor-ALPS makes a spatial prediction to identify which location to heat up. In this case, the HVAC system can be controlled to start increasing the temperature to the person's preferred temperature in advance of his arrival.

As we saw in the temporal-spatial results, we are not 100% accurate in our prediction, which is to be expected. There are two possible errors that can occur, which would affect the temperature control for the person's next location: 1) the spatial prediction was incorrect and the person is heading to a different office or 2) the spatial prediction was correct, but the temporal prediction was not. In case of a wrong spatial prediction the system would waste energy heating the wrong room. In case of a correct temporal prediction, heating will start on time or a little bit late, since we expect they may leave anytime in the next 30 minutes.

In the 11.5% of cases where an error occurs, two types are possible: predictions that the person will leave when they actually do not leave the current location in the next 30 minutes or the prediction that the person will stay when they actually will leave in the next 30 minutes. The Precision and Recall results for Indoor-ALPS and PPM have shown that Indoor-ALPS is more likely to make the latter error, while

PPM is more likely to make the former error. Thus the primary impact of temporal errors with Indoor-ALPS will be a delay in heating.

We believe that even though our algorithm sometimes starts heating too late, it will proactively reduce the energy consumption of a building. The errors Indoor-ALPS makes might affect the thermal comfort of the occupant, because the temperature did not reach the preferred temperature on time or in the correct room. The difference from the preferred temperature due to slight prediction inaccuracies might not even affect the thermal comfort much since the system only takes 10 to 15 minutes to recover and someone who was just walking is less likely to be cold. Looking at the large Recall difference between Indoor-ALPS and PPM we can expect that PPM would waste more energy than our algorithm and thus Indoor-ALPS would be better suited to reducing a building's energy footprint.

Following the same prediction model as described above another application our algorithm can support is an automated receptionist [26]. By predicting that an office occupant will remain in her office for a certain amount of time or longer, a colleague or visitor can be assured that she will be there when they arrive. Here the look-aheads are likely to be short, where Indoor-ALPS excels. By predicting if the building occupant will arrive at a new location in a specified amount of time (or less), we can assure the visitor he will meet the occupant if he arrives around that time.

Combining Temporal and Spatial Predictions with Oracles

Even though Indoor-ALPS already achieved a very high performance, we were interested to see whether we could further improve its performance by combining the temporal and spatial prediction (*i.e.*, allow the output of one prediction become the input for the other). To test this approach, we created two Oracle predictors. The Temporal-Oracle predictor has perfect knowledge about the next location of an occupant and uses that knowledge as an input feature in the prediction. The Spatial-Oracle, on the other hand, has perfect knowledge about if an occupant stays for the next n (n=10,…,90) minutes. If either Oracle with perfect knowledge (either spatially or temporally) results in a significantly improved performance, then we can try combining our imperfect predictors. However, when we evaluated the

use of these Oracles, we found that neither one provided an improvement over our original approach. Note that this result is true for the Augsburg dataset, and may still be worth investigating with a different dataset containing different types of mobility patterns.

4.5. Conclusion

The high performance of Indoor-ALPS highlights the importance of diverse spatiotemporal features for the prediction of future events. Although an important step, it still lacks the modeling power to represent complex spatiotemporal human behavior. Chapter 5 presents an algorithm that is built as a continuation of Indoor-ALPS and is capable of extracting this structure.

5. Exploring Spatiotemporal Structure for Indoor Location Prediction

The previous section showed us the importance of using diverse spatiotemporal features to model the complexities of a person's daily, weekly, or monthly routine. Even though Indoor-ALPS was an important step towards a viable indoor location prediction algorithm it still lacks the descriptive power to model complex spatiotemporal structure. For example, let us consider the situation of an office worker with a schedule like the hypothetical days shown in Figure 11 and Figure 12.

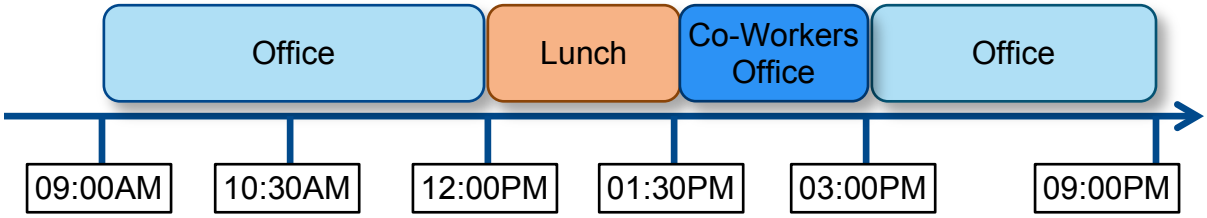

Figure 11 Hypothetical Thursday

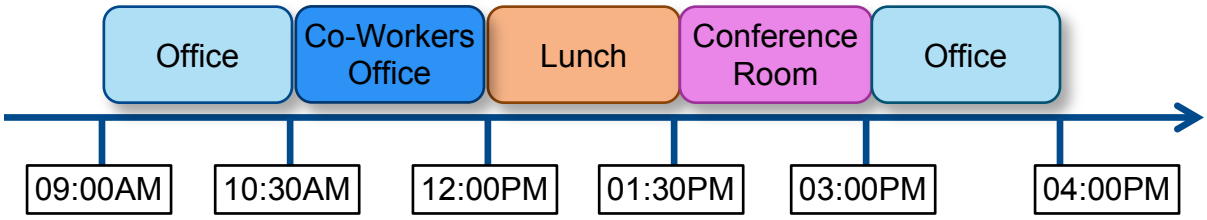

Figure 12 Hypothetical Friday

The occupant arrives in their office at 9:00AM in the morning and goes through a succession of locations before leaving their office at 9:00PM on Thursday and 4:00PM in the afternoon on Friday. Let us assume at 3:15PM on Friday we task an algorithm to make a prediction of the person's location one hour later, at 4:15PM. Indoor-ALPS would base its decision on features such as the current location, the arrival time in the morning, the current day of the week, or the previous two locations. This approach works well if a person's routine can be described through these features, but it is not capable of modeling more complex structure. For

example what if the person's departure time from work is not only dependent on what happened immediately before the current location, but also on events from the previous day. The fact that the occupant stayed late until 9:00PM on Thursday could be an indicator when the person will leave in the afternoon. Another complicating factor for a person's routine could be the temporal distance between two events. An event that happened the previous day might very well affect events that happen during the current day.

A person's daily behavior is partially defined by an underlying structure that consists of patterns of events. These patterns, which are formed consciously and subconsciously, allow us to navigate our busy life. They help families to coordinate their joint schedules [16], define how we navigate known and unknown streets [76], or determine when we heat or cool our homes [38,62]. Patterns that reoccur consistently form our routines. Routines, especially spatiotemporal ones, were studied previously to analyze human behavior such as application usage on desktop computers [10] or also in natural domains such as nature [50], geoscience [41], geophysics [63], meteorology [32], or embryology [64]. Understanding of human behavior based spatiotemporal routines can be used to study criminal behavior [14] or support family coordination [16]. A thorough description of a person's spatiotemporal structure can also allow us to explain the outcome of a predictive approach to the user and increase the intelligibility of a predictive algorithm. Studies have shown that this will impact user trust of the system [45]. In order to be able to model a person's complex spatiotemporal structure, this dissertation explores the use of *frequent-pattern trees* [29]. We use the resulting trees for prediction of future spatiotemporal events by extending the definition of frequent-patterns trees to include the temporal nature of spatiotemporal patterns by describing how events in a tree occur over time. The remainder of this section describes the behavior-pattern extraction algorithm, the complexity involved in extracting frequent-pattern trees from spatiotemporal data, and lastly an algorithm to use the extracted conditional frequent-pattern trees for prediction of spatiotemporal events.

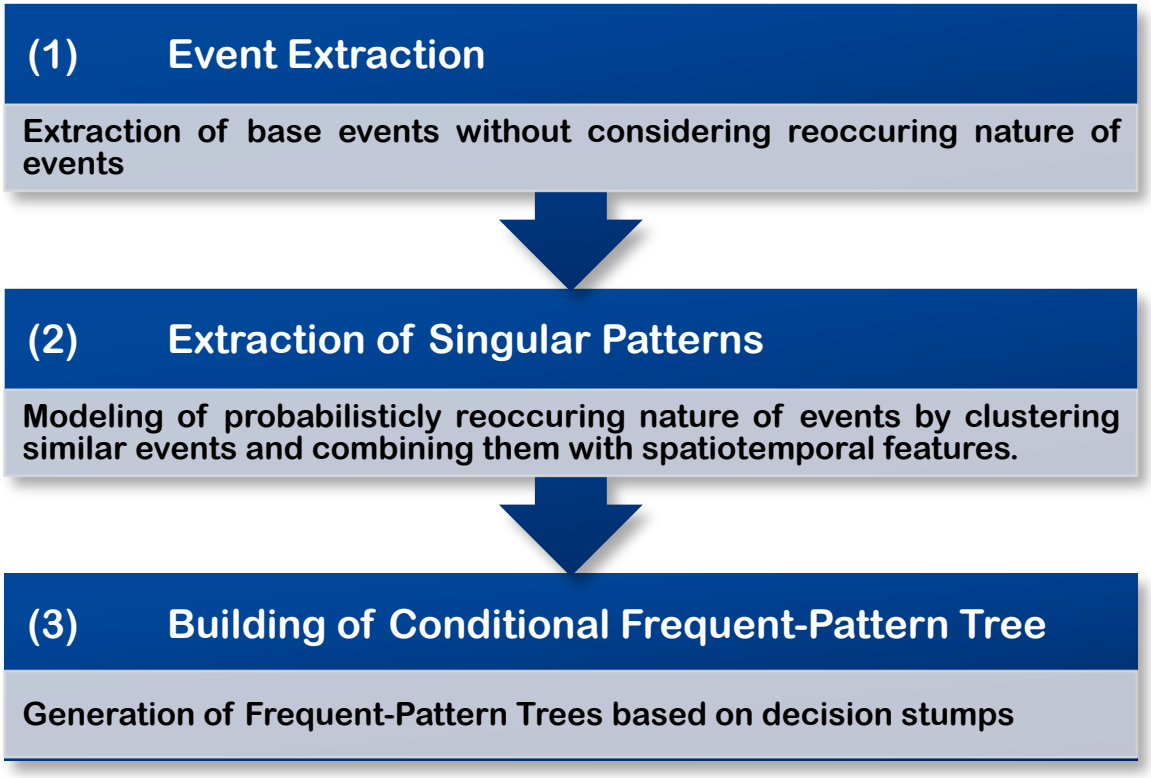

Figure 13 Algorithm Overview

5.1. Behavior-Pattern Extraction Algorithm

In order to create an algorithm that describes a person's spatiotemporal structure and predicts future spatiotemporal events we make the following assumptions: (1) a person's daily spatiotemporal structure consists of frequent and infrequent regular events and pattern of events, (2) spatiotemporal events are dependent on a wide-range of contextual features (*e.g.*, day of the week, time of day, previous visited locations, etc.), and (3) the occurrence of a future spatiotemporal event is dependent on the current event as well as events that previously occurred (*e.g.*, during the same or previous day/week). These assumptions are empirically derived from our own human experience. A person's schedule is not static, but a combination of numerous events. These events can occur with high (*e.g.*, a person arriving in their office at 8:30AM every workday) or low frequency (*e.g.*, a weekly meeting that only happens on Friday's), experience changes over time, and even be subject to seasonal shifts. Furthermore events can influence each other over longer temporal windows, for example, a person leaving their office at 10:00PM the previous day can influence their arrival in the office the next day.

To capture all of these varying assumptions we introduce ABC-Pattern-Extract (**A**ctivity, **B**ehavior and **C**ontext Sensitive **Pattern** Tree **Extract**ion), a pattern extraction algorithm based on Frequent-Pattern Trees [29]. Han et al. introduced Frequent-Pattern Trees as a compact data structure to capture ordered, frequent sets of items. The associated mining algorithm iteratively grows frequent conditional trees starting with a tree depth of one. Their results show that the resulting tree representations of a dataset are compact and computationally faster to generate than the popular Apriori approach. In addition to these attributes Frequent-Pattern Trees also allow us to consider diverse contextual features, which makes the algorithm *flexible* enough to explore not only patterns of individual behavior, but also relationships between contextual features such as people or spaces. Using Frequent-Pattern Trees as a basis, ABC-Pattern-Extract works in several stages as shown in Figure 13.

5.1.1. Event Extraction

ABC-Pattern-Extract models spatiotemporal events *E* as triples of location, time of day, and action. Throughout the algorithm we consider the following actions: transitioning to a location, transitioning from a location, staying at a location, first arrival of the day at a location, and last departure from a location for a day. It should be noted that this event description can be modified to model different behavior, which will allow the algorithm to be applied in different situations. For example it would be possible to also include a user field in this event description in order to model behavior that is affected by more than one user. The events extracted in this first step do not capture any notion of reoccurrence. Every event is treated as unique in the dataset. The extraction algorithm traverses the provided dataset and labels every data point with a unique event ID. Each event ID corresponds to an event in the format given above.

5.1.2. Extraction of Singular Patterns

As mentioned above, events do not capture the probabilistically reoccurring nature of a person's spatiotemporal structure. Thus, they only describe discrete moments in time. For that reason, ABC-Pattern-Extract models the reoccurring nature of events in the form of **Singular Patterns** (*i.e.,* patterns of length 1). Each Singular

Pattern models one class of events under varying spatiotemporal features. This subsection explains how ABC-Pattern-Extract creates Singular Patterns from the list of discrete events.

Event Clustering

To derive the aforementioned classes of events from the totality of events extracted in Step 1, we apply nearest neighbor clustering using the time component in each event as the cluster representative. Clustering is a necessary step due to the variability in human behavior. For example, let us assume a pattern that describes the event “Person arrives in office at 8:30AM every day”. To calculate the occurrence probability of this pattern, we need to determine which events are instances of such a pattern. But since human behavior is not static, multiple events can be instances of this pattern. For example a person arriving at 8:20, 8:25, 8:35, or 8:40 might all be instances of the pattern mentioned before. Clustering of the events based on the temporal component allows us to determine which event is an instance of a particular pattern. For the purpose of this clustering we define the following term:

> **Definition**: *Instance of Singular Pattern*
>
> An event *E* is a new instance of a Singular Pattern *P* if and only if it describes the same location and action and if the Time of Day (ToD) is not more than *m* minutes different from the mean ToD of all current instances of *P*.

Following this definition, the membership of an event E to a pattern P depends on a parameter m. This m is variable by user and even time of day. Intuitively the variance of the latter can be split up into two categories: the departure and arrival from the office building, for which we expect a higher degree of variance, and the transition time between locations during the day, for which we expect a lower degree of variance due to interactions between individuals. To account for these two cases we define two thresholds for the cluster mean, an *outer threshold* for patterns that are defined before 10:00AM and after 04:00PM and an *inner threshold* for patterns defined between 10:00AM and 04:00PM. In order to

determine the correct value for m, we need to evaluate the quality of the clustering. In our case a criterion of a good clustering is the reduction of the number of short clusters (clusters with less than five instances), since they also result in a low probability for the corresponding pattern. A very fine-grained clustering of events is counterproductive for the probabilistic modeling of spatiotemporal structure. Since this is an unsupervised clustering problem (there is no prior knowledge about the number of clusters), we employ the Dunn Index as an evaluation scheme. The evaluation of an unsupervised clustering is a challenging problem since by the nature of the problem no ground truth exist. We picked the Dunn-Index for our approach since the only real drawback is the computational complexity and we do not expect to have enough clusters for this to become a problem. The Dunn-Index tries to minimize the within cluster distance and maximize the between cluster distance, which means that for a given set of clusterings a high Dunn-Index indicates a better clustering. The Dunn Index (DI) is defined as:

$$DI_m = \frac{\min\limits_{1 \le i < j \le m} \delta(C_i, C_j)}{\max\limits_{1 \le k \le m} \Delta_k}$$

- δ (C_i,C_j) is the distance between cluster C_i and C_j
- Δ_k is the diameter of cluster C_k calculated as the distance from all points to the mean

We create a new clustering of events for a particular set of *outer* and *inner thresholds* (with threshold values between five minutes and 35 minutes in 5-minute steps we get 128 sets of *outer* and *inner thresholds* and subsequently 128 different clusterings). For each clustering the Dunn Index is calculated. The goal of these individual clusterings is to determine the “best” set of *outer* and *inner thresholds*, *i.e.*, the number of minutes m a new instance can be different from the mean of a temporal cluster. For example if the mean of a cluster is 8:30AM and a new instance is described for 8:45AM than the instance belongs to the cluster if m is greater or equal than 15 minutes. After a clustering is calculated for each pair of thresholds the algorithm orders the cluster results by their Dunn Index and picks the clusterings with the five highest index values. In some cases the Dunn Indices

for the five best clusterings are very similar. To still decide on a set of outer-inner thresholds (keeping in mind that the criterion for a good clustering is a reduction in the number of short clusters), we also calculate the ratio of short clusters to the overall number of clusters as a cluster coefficient and pick the *outer* and *inner threshold* set out of the five Dunn Index results that has the lowest cluster coefficient. We do not use the number of short clusters as a decision criteria directly to prevent clusterings from being picked that create a low number of short clusters by assigning the majority of data points into a small number of large clusters (which have a very large diameter because of it) and the rest into only a few short clusters.

Singular Pattern Definition

After each event is assigned to one cluster (*i.e.*, one cluster holds multiple events) ABC-Pattern-Extract creates patterns of length one by combining a number of contextual features with each event cluster. For example let us assume an event cluster that contains events of the following nature: “Staying at office at 9AM” and furthermore a contextual feature: every Friday. By combining these two pieces we derive a pattern that describes the following situation: “Every Friday a person stays at their office at 9AM”. We refer to these patterns of length one as **Singular Patterns,** and they form the building blocks for our Frequent Pattern Trees. To formalize this as a definition:

> **Definition:** Singular Patterns
>
> A Singular Pattern is a combination of an event cluster with one contextual feature.

The contextual features that can be determined and derived from spatiotemporal data fall into three categories: temporal, spatial, and location *and* time dependent features. In addition we can also consider location- and time-independent features (e.g., current weather, road closures due to constructions) or even spatiotemporal features from a second person. Our current implementation of ABC-Pattern-Extract considers the following features, which are ranked by their specificity from least to most specific:

1. Daily
2. Week vs. Weekend indicator
3. Season
4. Day of week
5. n-th weekday of the month
6. Day of month
7. Day of year

We strived to make this list as exhaustive as possible so we can capture as many different situations as we might encounter in a person's spatiotemporal behavior, but this is by no means a complete list. Especially with the location- and time-independent data, which we did not model in the current implementation, we can imagine other contextual factors that might affect a person's spatiotemporal behavior. By describing the behavior through events with contextual conditions, this definition allows for extensions and adaptation to a specific problem domain. We expect the current definition to be good enough to support multiple different application domains, but with access to additional datasets it is possible to easily extend the current set of features.

Singular Pattern Frequency

Each Singular Pattern has an attached frequency of occurrence (i.e., the number of instances in the current history that belong to that Singular Pattern given the contextual features) and probability. The probability is calculated by dividing the frequency of occurrence by the theoretical maximum frequency of occurrence. This leads us to the following definitions:

$$Fr(SP) = \#\ of\ Instances\ belonging\ to\ SP$$

$$\Pr(SP) = \frac{Fr(SP)}{Fr_{max}(SP)}$$

(SP = Singular Pattern; $Fr_{max}(SP)$ = maximum theoretical frequency given contextual features of SP)

For example, let us assume a Singular Pattern P describes the following event: "Person stays at office at 10:30AM every day". If we encounter 31 days in our

dataset and P on 23 of these 31 days, we would assign a probability of occurrence of 23/31 or 74.2%. Each Singular Pattern gets assigned a unique identifier.

5.1.3. Building of Conditional Frequent-Pattern Trees

As we will see later in the result section, Singular Patterns alone are not enough to capture the spatiotemporal structure of human behavior. Spatiotemporal behavior is not only influenced by the current context (*e.g.*, time of day or day of the week), but also by events that happen earlier in the day or even the previous day or week. Modeling larger spatiotemporal structures that describe these more complex routines requires the extraction of longer patterns of events. Researchers have investigated the use of various algorithms to capture the structure found in event-based datasets. For example, Ryan et al. [61] explored the use of Association Rule Mining for prediction of indoor location events. Even though they reported some success on a self-collected dataset with an accuracy of 79%, the performance dropped to 56% when applied to the Augsburg Location Tracking dataset. They explained this drop as a result of the increased variability in the dataset. One important drawback of association rules is their inability to model the temporal structure found in spatiotemporal datasets. The members of a rule can appear in any order and over any time frame, which results in an inability to model the structure we see in such datasets.

Another class of algorithms is Markov Decision Processes (MDP), which were successfully used in the past to model a wide range of spatiotemporal problems. One very successful recent approach is PROCAB by Ziebart et al. [76]. Their algorithm used an MDP combined with inverse reinforcement learning and the principle of maximum entropy to model the behavior of drivers. The approach achieved a very high performance in trip endpoint prediction (~ 3km error) and turn prediction (93.2% accuracy). Even though the performance is very promising, PROCAB and MDP's in general are not useful for the modeling of indoor spatiotemporal behavior. PROCAB does not model the temporal context of indoor behavior and in fact including the temporal structure (an indicator how events happen in time: on the same day, over two previous days, during the same week, etc.) would increase the computational complexity by a large degree. Modeling the

temporal structure of events would require either to create a new MDP for each temporal condition (*e.g.,* same day, adjacent day, same week, etc.) or an extension of the current MDP to allow it to learn individual feature weights for each temporal condition. Both solutions would result in an increased computational complexity. In addition, MDPs make the Markov assumption that future states are only dependent on the current state and as highlighted earlier one of the core assumptions we make is that future states are highly dependent on previous events and that in fact a chain or pattern of events will allow us to predict the future.

One approach that will allow us to model both the sequential structure found in spatiotemporal behavior as well as the temporal nature are Frequent-Pattern Trees, which were introduced by Han et al. [29]. Frequent patterns play a vital role in many sequence mining problems and Frequent-Pattern Trees (FP-Trees) offer the promise of providing a compact representation of these patterns. ABC-Pattern-Extract is based on the FP-Tree growth algorithm and works in several stages by iteratively growing the depth of each tree starting from each Singular Pattern.

Singular Pattern List Filtering

The first step of ABC-Pattern-Extract is to create a reduced list of Singular Patterns by filtering out patterns that have a frequency lower than a frequency threshold. We did not use the Singular Pattern probability, since, depending on the contextual features, a probability can be very high even though the pattern is very infrequent. This especially happens with very specific features such as the indicator if a Friday is the fifth Friday of the month. The intuition behind a frequency threshold is that the frequency of longer patterns is limited by the smallest frequency of its sub-patterns. For example lets consider a pattern [A B] (a path in an FP-Tree) of length two that consists out of the Singular Patterns A and B. This pattern indicates that after A pattern B follows. The frequency of [A B] is bounded by the lowest frequency of the individual Singular Patterns.

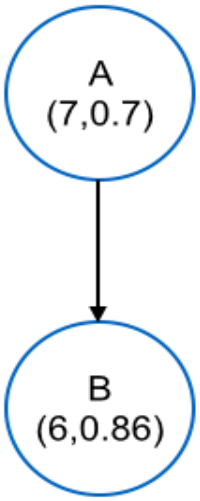

Figure 14 FP-Tree Example

$$Fr([A\ B]) \leq \min\{Fr(A), Fr(B)\}$$

This threshold allows us to adjust ABC-Pattern-Extract's sensitivity to irregular events. The higher this threshold the more Singular Patterns will be excluded, which has two effects: 1) the computational complexity goes down since less FP-Trees will be created and 2) the resulting behavior description becomes potentially less expressive. An ideal threshold would minimize the computational complexity (*i.e.,* create less FP-Trees) and maximize the expressiveness. For our experiments we picked four as the threshold. A deeper analysis of the effects of different thresholds on the extracted FP-Trees and their effects on the prediction of spatiotemporal events is out of scope of this dissertation and will be tackled in the future work.

The chosen frequency threshold results in a reduced list of Singular Patterns, which ABC-Pattern-Extract uses in order to create the FP-Trees (a directed, acyclic graph) of depth two by extracting all possible combinations of two unique Singular Patterns (*e.g.,* [A B], [A C], [B C], [C A], etc.). That way each Singular Pattern in the reduced list becomes the root for its own FP-Tree. The nodes of this tree are Singular Patterns and a path in the tree from the root to a node is called a pattern. This means a pattern consists of two or more Singular Patterns and the length of a pattern is determined by the number of Singular Patterns (*i.e.*, number of nodes in the tree) along the corresponding path in the tree. Furthermore, each Singular Pattern in a pattern has to be unique (meaning [A A] is not possible) and a pattern imposes a temporal order. Each node in a FP-Tree has two associated values: the frequency of the pattern that ends in that node and the conditional probability for the node given the sub-pattern without that node (for example the conditional probability for a pattern [A B] would be Pr(B|A)). To exemplify this process, let us consider the following Singular Patterns: A ("Person stays in office at 9AM on Fridays") and B ("Person leaves office at 5PM on Fridays") with Fr(A) = 7 and Fr(B) = 8. Let us further assume that the training dataset contains 10 Fridays, which would lead to Pr(A) = 0.7 and Pr(B) = 0.8. Taking A as the root, ABC-Pattern-Extract would create the tree shown in Figure 14.

As we see in the example, the frequency of [A B] is 6, meaning for the seven instances of A we observe B for six of them. This also leads directly to the conditional probability of B occurring after observing A, which is 6/7 or 85.7%. To model human spatiotemporal routine we also need to make an extension to the FP-Tree definition by including a *temporal condition* under which events in a pattern occur. Our daily life shows us that events can occur over large or small temporal frames, for example two events can occur on the same day or also across two adjacent days.

> **Definition:** *Temporal Condition*
>
> For patterns of length two or greater, the associated temporal condition describes how instances of the Singular Patterns, which constitute that pattern, occur over time. One pattern can be defined for multiple different temporal conditions with its own frequency and conditional probability.

In order to capture how events occur over time the following temporal conditions are defined: same day/week/month/year, next adjacent day/week, and next two adjacent week/month. We decided to model the temporal condition in this predefined way because learning the mean temporal distance over which a sequence of events can occur is computationally prohibitively expensive. Given these temporal conditions we can see that we would need to learn the frequency and probability for [A B] for several temporal conditions (same day, week, month, and year). But we also see that it is impossible to learn these values for all conditions since both A and B are defined for Fridays and thus it is impossible to define the frequency or probability for any of the adjacent temporal conditions. Our definition allows the algorithm to mix Singular Patterns with different temporal features (*e.g.*, one is defined for every day and the other only for Fridays). Another important aspect is that a pattern also implies a temporal order. For example, a pattern [A B] implies that instances belonging to A have to occur before instances that belong to B. It is to note that a temporal condition is assigned to a pattern of length two or greater and is not to be confused with temporal features, which are

assigned to Singular Patterns. The purpose of the temporal condition is to describe how two or more events occur in time.

After ABC-Pattern-Extract created all possible trees with a maximum depth of two (i.e., pattern length of two), it will iteratively grow these trees. In order to do this ABC-Pattern-Extract extends a path in a FP-Tree by adding every possible Singular Pattern in the reduced list to the path. Since this potentially has exponential growth, we also limit the expansion by limiting it to paths in the tree that have a frequency of greater than four using the same reasoning as before. The expansion is stopped if the information content (as defined by the self-information of a pattern) of the collection of patterns over the last three pattern lengths is monotonously decreasing. The self-information of a pattern P is commonly defined as:

$$IC(P) = -\log \Pr(P)$$

Since this is an additive measure we can calculate the information content of a collection of patterns as the sum of the self-information for each pattern. The intuition behind the this criterion is that if there are not enough new patterns being created, it is not worthwhile to extend the pattern length since it is computationally expensive.

5.1.4. Pattern Extraction Results

In order to evaluate ABC-Pattern-Extract we extracted the FP-Trees from the Gates-Hillman-Center dataset for Sept. ‘11 and Oct. ’11 and afterwards tested the

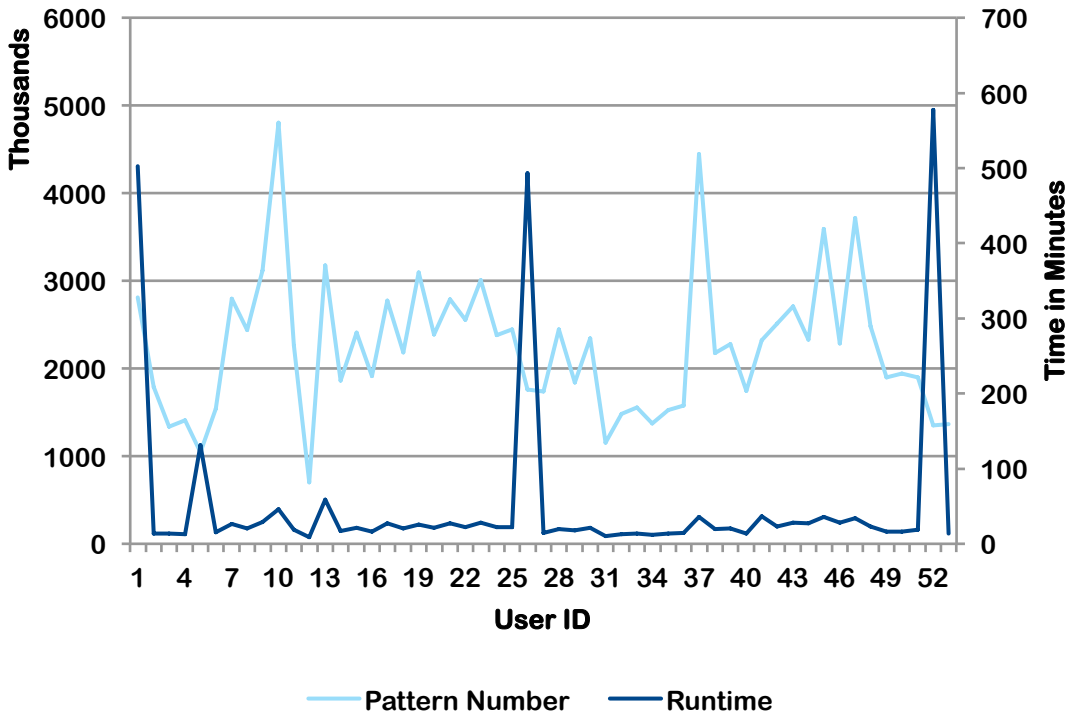

Figure 15 ABC-Pattern-Extract Number of Patterns vs. Runtime

prediction algorithm we describe in the next section on Nov. ’11 and Dec. ’11. We decided to limit the analysis on these four months for two reasons: one they overlap with the Fall semester at Carnegie Mellon University and thus have less seasonal changes, which allows us to scope the pattern extraction by preventing the algorithm from using all spatiotemporal feature and temporal conditions (the final list is shown below). In a university setting, the spatiotemporal routine of building occupants is highly dependent on the academic semester: changes in class schedules impacts all primary groups of occupants (faculty, student, staff). For example while a faculty might be in their office most of the day during the summer semester, this will change during the fall and spring semester due to classes a faculty offers. Secondly, extending the analysis length over a longer time frame would require the implementation of an update algorithm that creates new FP-Trees and updates current ones. Deciding when to update a spatiotemporal

behavior model is out of scope for this dissertation and will be tackled as part of the future work. The final set of spatiotemporal features that were used for this analysis was:

- location,
- time of day,
- day of week,
- week vs. weekend indicator,
- n-th weekday of the month,
- day of month.

We omitted the use of the day of year and season feature since the length of the training set would result in not enough variance in the feature values. Similarly we only extracted frequency and probability values for the same day/week, adjacent day/week, and two adjacent day/week temporal conditions. In addition we stopped the extraction process when the number of calculations exceeded 100,000,000 (a computational stop criterion that can be relaxed or even removed by running the code on a cluster). Due to the complexity of the computation and the availability of computational resources, we limited our analysis to 1/5 of the rooms in the dataset by randomly selecting 53 offices (a large enough sample to draw statistical comparisons). Before the selection process we made sure that all the offices have roughly the same number of data points and excluded extreme offices (rooms with either below 5% or over 95% average occupancy). We summarize the following statistics about ABC-Pattern-Extract: number of Singular Patterns, number of patterns in the resulting FP-Trees, average number of FP-Trees, average FP-Tree depth, and runtime in minutes. The limitation we put in place allow ABC-Pattern-Extract to be run on a machine with a 2.6GHz Intel i7 processor, 16GB of DDR3 RAM, and a 500GB Solid State Drive. In addition, since all the code was implemented in Python, we used Python's process pool to parallelize the code execution.

On average ABC-Pattern-Extract extracted 2,242,892 patterns (SD=799,869.9) per user and ran for 52.5 minutes. Three users took significantly longer to run,

which we believe to be outliers, since the number of patterns for these three users fall within one standard deviation above and below the mean. Removing these outliers gives us a runtime of approximately 24 minutes. As mentioned before, the extraction was performed on a single standard laptop, which means that this time estimate can be even further improved by utilizing a distributed system. Adding additional contextual features will increase the extraction time, but given the short extraction time for our evaluation, we believe that this estimate will not significantly increase. Since the extraction time on a standard laptop is so short it will also be possible to train the initial model on a modern smartphone, which removes the need for a dedicated server architecture. As a point of comparison, the latest Samsung smart phone, Galaxy S6, is equipped with a 2.1 GHz ARM Cortex Quad-core, 4GB RAM, and an extensible 64GB storage. As the approach should be capable of being executed on a smart phone, we feel comfortable in claiming that our approach is time-efficient. The pattern extraction was based on an average of 7194 Singular Patterns (SD=802.7). Overall ABC-Pattern-Extract created on average 1011.6 FP-Trees (SD=154.9), which had an average depth of 5.3 (SD=1.14). The latter means that the average pattern length is also 5.3.

To understand whether our algorithm is efficient in terms of disk space required to store the results, we analyze the space requirements. The space complexity of our approach depends on the number of extracted patterns and the space complexity of storing a single pattern. All patterns were stored in MySQL with a mixture of integer, double, and text fields (each with their own space constraints). In total our algorithm saves four integer values (each 4 Bytes), two double values (each 8 Bytes), and three text values (length dependent on the length of the string). The space complexity for the double and integer values is easily computed as 32 Bytes. The length of the text values depends on the pattern that is being saved: one field saves the pattern members, one the pattern condition, and one the temporal condition. The temporal condition has at most 22 Bytes and the two other fields are bounded by the maximum tree depth (pattern length). Taking the average values above it takes on average 105.7 Bytes to store one pattern. Given this, it would take on average 226 MByte to store the average number of patterns

mentioned above. This would make storing of these patterns on a mobile device feasible and could allow the prediction algorithm to run independently of a server. One of the biggest challenges of mobile computing (computing using devices such as smart phones) is the resource constraint in terms of storage and computing power. Our algorithm is efficient enough to run the extraction and storage of patterns on such a device. This analysis answer the first question we proposed in the introduction: **Given complexity of modeling human spatiotemporal behavior and reoccurring events how can we compute conditional frequent-pattern trees for both in a space and time efficient manner (Q1).**

5.2. Using Behavior Patterns For Indoor Location Prediction

As mentioned earlier the extracted FP-Trees can be used to either explain behavior to users in an effort to allow them to discover interesting insights about their data or we can predict future spatiotemporal events. We developed ABC-Pattern-Predict (**A**ctivity, **B**ehavior and **C**ontext Sensitive **Pattern** Tree **Predict**ion), an indoor location prediction algorithm on the basis of FP-Trees extracted by ABC-Pattern-Extract. Our algorithm has two modes of operation: predictions based on Singular Patterns and predictions based on the full FP-Trees. Each mode has its own set of advantages and disadvantages: making predictions based Singular Patterns (patterns that describe single reoccurring events given contextual features) alone is fast and independent of a temporal look-ahead window, but it impacts the prediction performance since it makes in the moment decisions without knowledge about events that happened earlier. The independence from a look-ahead window also means that the algorithm can make predictions for any time of the day, essentially allowing it to create a daily or weekly schedule ahead of time. The second mode for ABC-Pattern-Predict allows the algorithm to make very precise predictions by using the full range of prior events, but it comes with the added cost of increased runtime and a dependence on a look-ahead window. Depending on the situation, a user of ABC-Pattern-Predict has to decide which mode to choose.

It is to note that we performed all the initial experiments (that is trying different approaches to use the extracted FP-Trees) on a separately collected 18 user and

three month long dataset. The data was collected using the same occupancy sensor infrastructure over a period of three month on March 2015 through May 2015. As indicated above the final validation was performed on a completely separate dataset collected earlier.

5.2.1. Mode 1: Prediction Based on Singular Patterns

Making predictions based on Singular Patterns is a very straightforward process. For a time step t (given as a discrete 5-minute time chunk), ABC-Pattern-Predict picks Singular Patterns that match the current time of day and temporal features and orders by their probability. This list of potential Singular Patterns contains patterns for all possible locations that were encountered during time step before. From the list of potential patterns it chooses all patterns that are less than 5% away from the pattern with the highest probability. For some patterns the probability is either very close to each other or even the same. Since this increases the uncertainty, we consider all patterns that are very close to each other as possible candidates and choose the pattern out of that list with the most specific feature value. As mentioned earlier the feature values are ranked by their specificity (*e.g.*, daily patterns happen every day, but day of week patterns only happen on a very specific day), which allows us to also rank the patterns in the result set. To give an example of this process, let us assume the situation shown in the beginning of this chapter (see Figure 12). We want to make a prediction for 2PM and determine that the following Singular Patterns are applicable for this situation: A (“Person stays in office at 2PM on weekdays” – P(A) = 90%), B (“Person stays in conference room at 2PM on Fridays” – P(B) = 88%), and C (“Person stays in office at 2PM daily” – P(C) = 75%). As we can see the two most likely patterns are A and B, but since B describes the more specific temporal feature it would be picked even though the probability is slightly lower.

5.2.2. Mode 2: Prediction Based on FP-Trees

The prediction of future spatiotemporal events based on all FP-Trees depends on events that occur prior to the current situation. Intuitively the spatiotemporal structure of a person consists out of three base-type of events: irregular events (events that can be either highly infrequent events or anomalies), infrequent

regular events, or frequent regular events. Since it is hard to make predictions based on irregular events, due to its non-deterministic nature, our algorithm, at its core, models two types of events: frequent regular events (events that have a high probability of occurring even using Mode 1 as predictor) and infrequent regular events (events that are regular, but only appear in conjunction with other events). We use two different algorithmic approaches for these events, a Maximum Likelihood Estimator models infrequent regular events, while a weighted Maximum Likelihood Estimator models frequent regular events by multiplying (*i.e.*, weighing) the conditional probability of a pattern with the probability of the target Singular Pattern. For a time step t, ABC-Pattern-Predict uses the look-ahead window length ε (a parameter that needs to be provided by the application domain developer) to make a prediction for time step t + ε. For each time step, our algorithm makes two independent decisions using the MLE and wMLE and decides which result to choose based on the frequency of the Singular Patterns that describe the events that occur at time step t + ε. In the beginning, the algorithm extracts all possible patterns that are defined for time step t + ε as a list of potential patterns. A pattern is defined for a time step t + ε if and only if the Singular Pattern at the end of the pattern is defined for time step t + ε. For example if B describes events happening at 12PM and C describes events happening at 4PM and we want to make a prediction for 12PM, then patterns that end on C are not defined for the current situation and are subsequently not picked. In order to decide which pattern to pick, it performs several steps, which are summarized in the following list and explained below:

1. Multiply the probability of each pattern with a temporal interval (same day, previous day, etc.) dependent weight
2. For each temporal interval, determine the pattern with the highest probability (Maximum Likelihood Estimate - MLE) across all possible locations and the pattern with the highest weighted probability (weighted Maximum Likelihood Estimate - wMLE) across all locations; essentially creating two temporal interval lists, one for the MLE and one for wMLE case; the number of entries

in each list depends on the number of temporal intervals that were considered

3. For both lists, independently pick the pattern with the highest probability
4. Determining the frequency of the Singular Patterns SP_{MLE} and SP_{wMLE} at the end of each pattern (this SP describes events for time step t + ε) and decide which pattern to pick based on:

$$RP(\mathrm{t} + \varepsilon) = \begin{cases} SP_{MLE}, & Fr(SP_{MLE}) \,/\, Fr(SP_{wMLE}) < 0.5 \\ SP_{wMLE}, & Fr(SP_{MLE}) \,/\, Fr(SP_{wMLE}) \geq 0.5 \end{cases}$$

($RP(\mathrm{t} + \varepsilon)$ = Result pattern at time step $\mathrm{t} + \varepsilon$; SP_{MLE} and SP_{wMLE} = Singular Pattern for MLE and wMLE result; $Fr(SP_{MLE})$ = Frequency of SP_{MLE})

Explanation of Steps

As we can see, ABC-Pattern-Predict works in several interlocking steps. Since it is unknown how important each temporal interval is for the prediction, we allow ABC-Pattern-Predict to weigh each temporal interval in **Step 1**. The intuition is that our algorithm learns over time how important each temporal condition is given the spatiotemporal structure of the user. For example if most of the spatiotemporal structure is contained in same-day patterns, our algorithm will learn a higher weight for that condition. To that end, the algorithm incrementally learns [27] weights for each temporal condition (independently for the MLE and wMLE prediction) starting with a weight vector that is initialized with 1's at the beginning of the validation dataset. This process happens during the prediction and not the training step. The weights are updated after each prediction. ABC-Pattern-Predict adds and subtracts a value epsilon (in our experiments set to 0.01) to the individual weights if a prediction was correct or incorrect, respectively. These weights are multiplied with the conditional probability for each pattern in Step 1.

In **Step 2** we split our prediction into two independent predictions using an MLE and weighted MLE to allow ABC-Pattern-Predict to react to the two types of events mentioned previously. In this process the MLE always picks the pattern with the highest conditional probability. Depending on the pattern, this probability can be very high even if the individual events described by the pattern are very infrequent.

For example let us consider a pattern [A B], where A occurred before and B is a potential target pattern for time step t + ε. A and B can be very infrequent, with low individual probabilities, but in 90% of the times if we observe A we also observe B. This means the conditional probability is very high and the MLE would pick up on this and chose [A B]. In this way the MLE allows us to pick infrequent regular events. The weighted MLE on the other hand would multiply the conditional probability with the probability of B occurring. Since we said before that B has a low probability we would also observe a low probability by weighing Pr(B|A) with P(B). The weighted MLE would most likely not pick [A B] as a potential result pattern and instead pick a result pattern with a higher wMLE value. Using all potential patterns to make a decision about the location at a given timestamp proved to be unsuccessful. Using the experimental dataset mentioned in the beginning of this section, we explored the use of calculating the mean and median probability across all patterns belonging to a particular location and deciding on the target location based on the highest mean/median probability. Unfortunately the results were significantly lower than our approach.

Since we use a Maximum Likelihood Estimator (with or without weight), we choose the pattern with the highest probability in **Step 3**. To make the final decision in **Step 4** between the MLE and wMLE result, we use the equation shown above to calculate the relative frequency of the target Singular Patterns described by the MLE and wMLE result patterns. This equations is calculating the result pattern, RP(t + ε), at time step t + ε by calculating the relative frequency $Fr(SP_{MLE})/FR(SP_{wMLE})$ for the target Singular Pattern, SP_{MLE}, in the MLE result and the target Singular Pattern, SP_{wMLE}, in the wMLE result. If this relative frequency is greater or equal to 0.5 SP_{wMLE} will be picked otherwise SP_{MLE} will be picked. We chose not to use a fixed frequency threshold since it is unclear what constitutes a frequent event for a given user and situation. Exploring this question is out of scope for this document and will be tackled in future work. Instead we defined the relative frequency between the MLE and wMLE result. The MLE result will be picked if the frequency of the target MLE Singular Pattern is less than half of the frequency of the target wMLE Singular Pattern. For example let us assume we

have the following two patterns after Step 3: [A B] (MLE result pattern) and [A C] (wMLE result pattern). To make a decision between these two patterns we calculate the relative frequency of B and C, which are both potential patterns that describe events at time step t + ε. If the relative frequency is above 0.5, ABC-Pattern-Predict picks C as result pattern or B otherwise. Meaning if the frequency of B is 20 and the frequency of C is 21 than the relative frequency is 0.48 and thus B will be picked. If C has a frequency of 20 or below C would be picked. The intuition behind this process, is that the MLE has a higher chance of producing the correct result for low frequency patterns (an intuition that held true as our initial experiments and the results showed) and the wMLE produces a better result for high frequency patterns.

Since it takes several steps to determine the final result pattern (including update steps for weights), it takes several seconds to make a prediction for each time step. In a real world scenario, this characteristic is perfectly acceptable since a prediction is not made in real time every second, but instead every five minutes. The complexity of the algorithm depends on the length n of the validation set. Extracting the possible Singular Patterns that apply to a current situation requires the traversal of all data points that were encountered so far, which means for an input size of n the algorithm makes n-1 comparisons. This means the algorithm makes a total of n*(n-1) comparisons. In addition to the pattern extraction step, the algorithms also needs to process all m potential patterns, which happens three times congruent with Steps 1 through 3. This results in an overall time complexity of $O(n^2 – n + 3*m)$. Since the number of target patterns m is much smaller than n^2 we can estimate a worst case complexity of $O(n^2)$. Given this time complexity ABC-Pattern-Predict has enough time to make predictions for multiple users every minute, even using off-the-shelf hardware.

6. Evaluation

In order to evaluate ABC-Pattern-Predict we compared the performance of our algorithm (using Mode 2) against two prior algorithms (PPM and PreHeat) as well as a ZeroR baseline predictor. We report the results for the following measures: accuracy, precision, recall, and kappa. In addition we report on two different time frames: all-day results and work-time (Mon-Fri 8AM to 6PM) results. Since the results of PPM and the ZeroR predictor are significantly weaker than the ABC-Pattern-Predict and PreHeat results we decided to limit the deeper investigation of precision, recall, and kappa to latter two algorithms. Figure 16 through 23 shows these results.

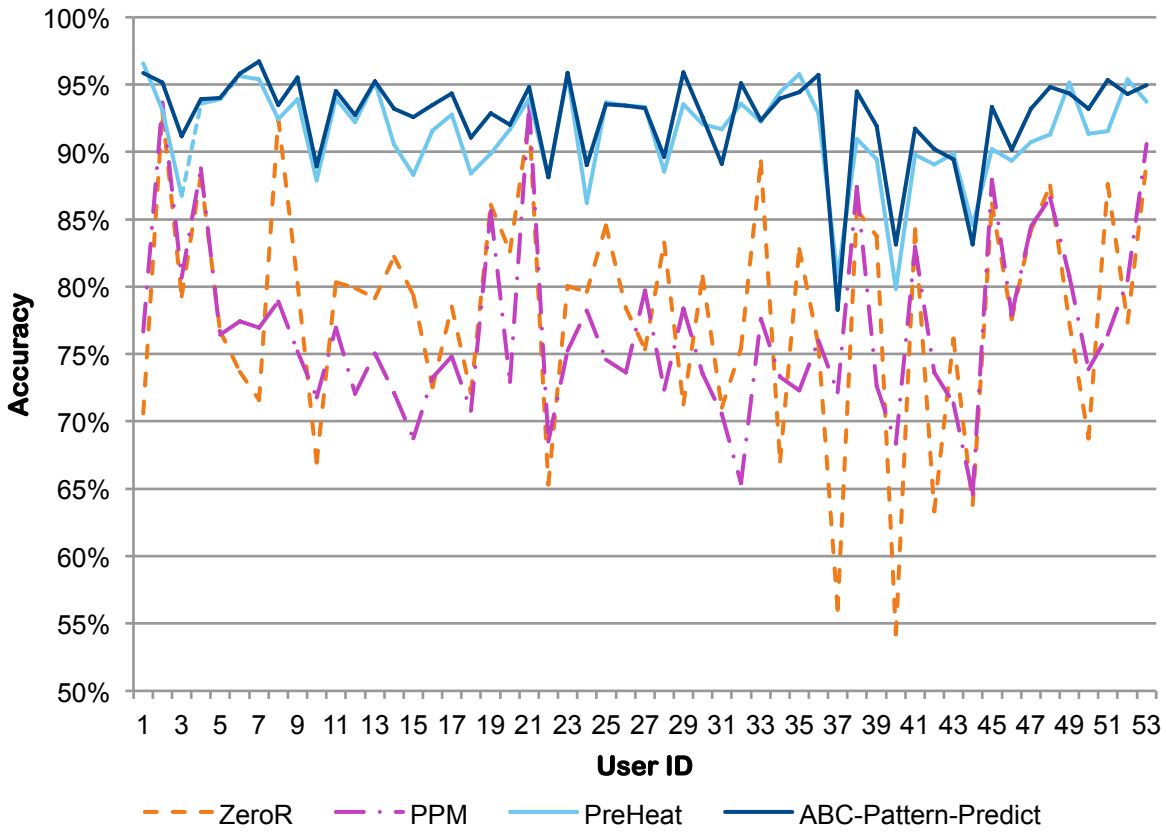

Figure 16 All-Day Accuracy

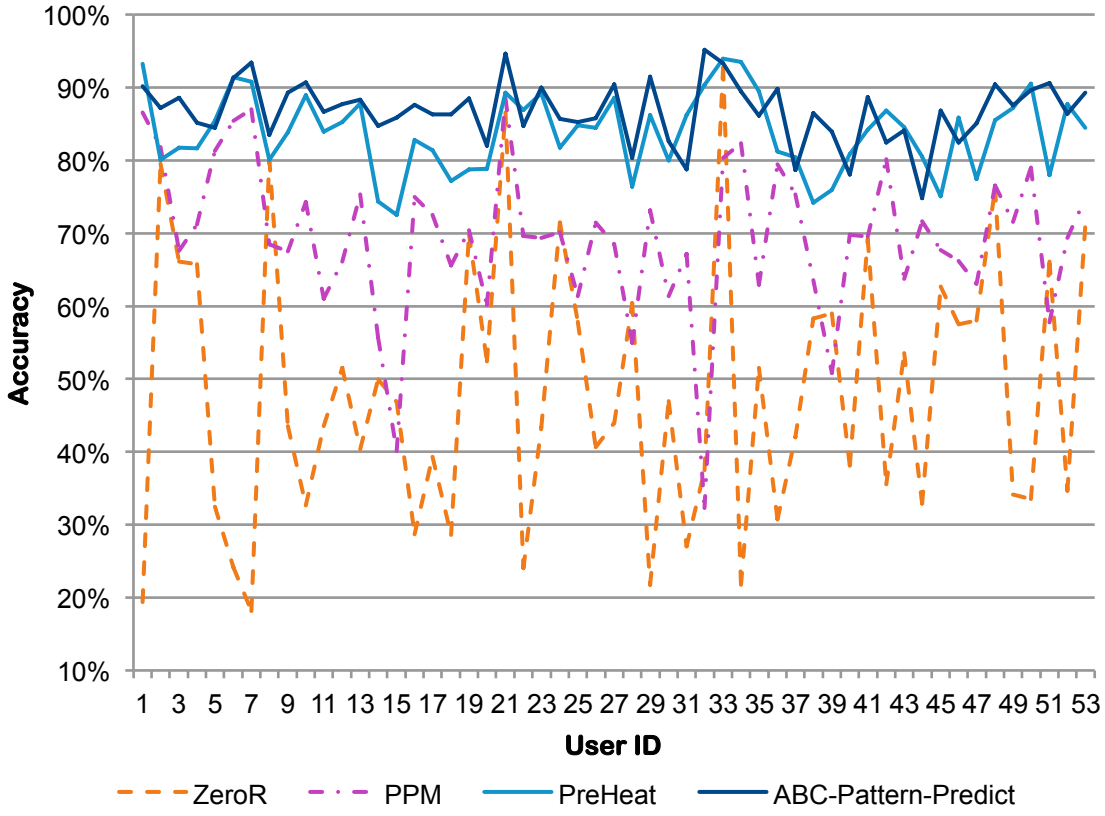

Figure 17 Accuracy Mon-Fri 8 AM to 6 PM

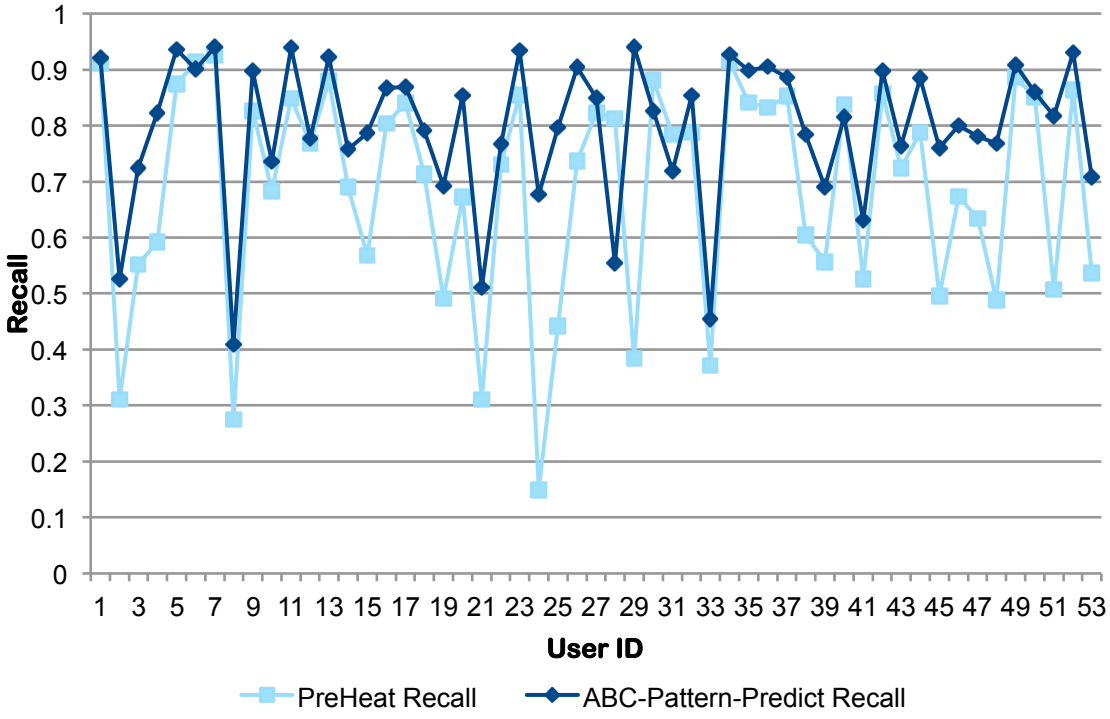

Figure 18 Recall All-Day

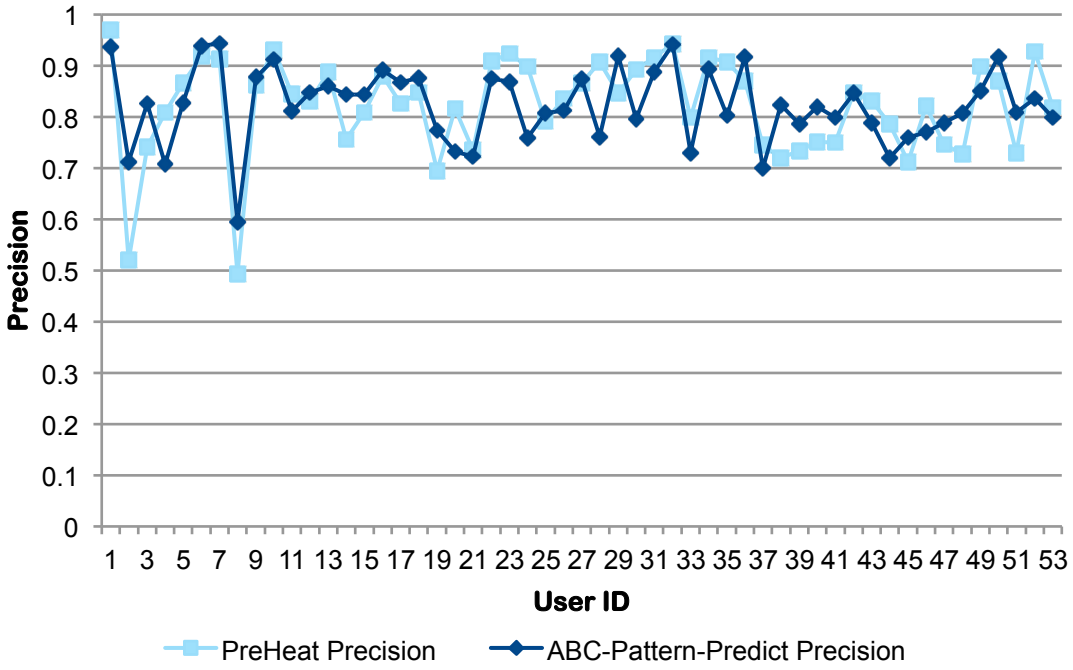

Figure 19 Precision All-Day

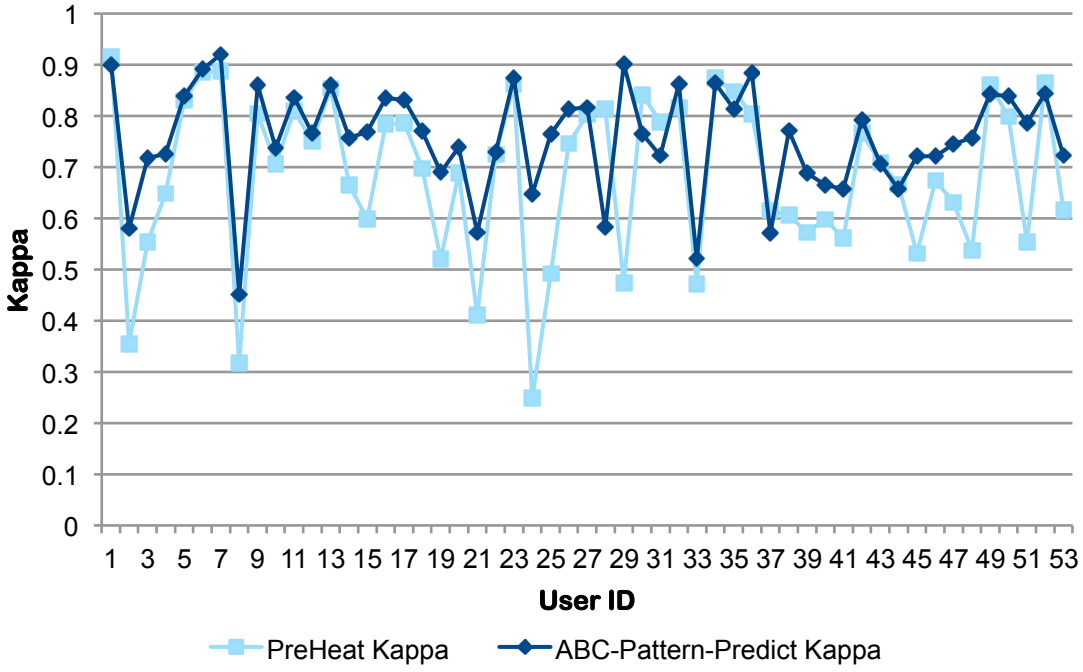

Figure 20 Kappa All-Day

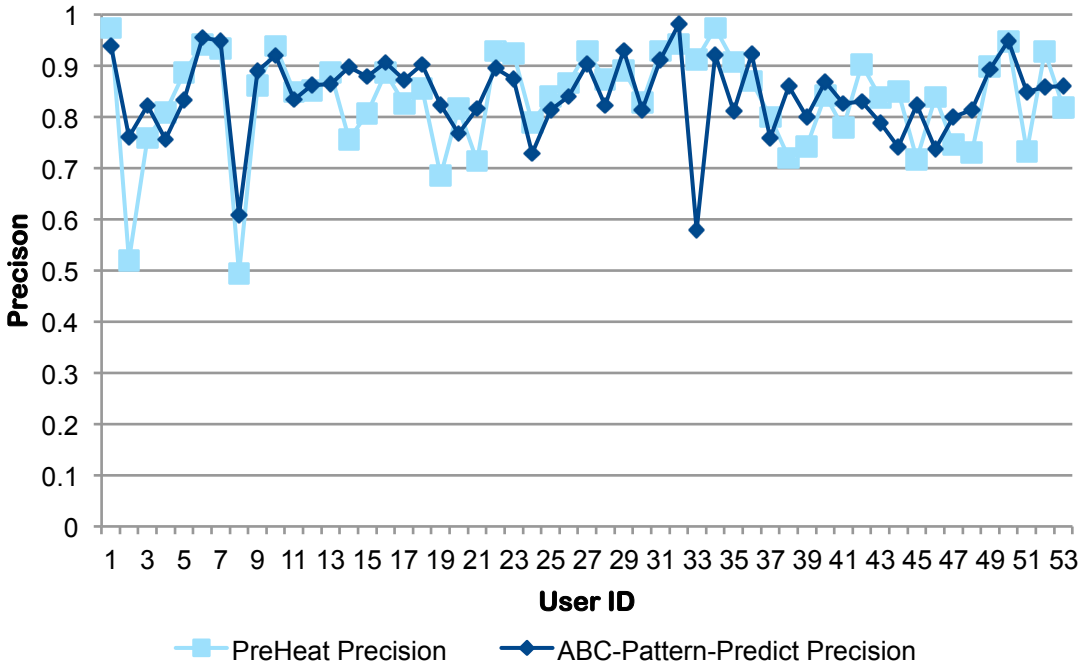

Figure 21 Precision Work-Time 8AM - 6PM

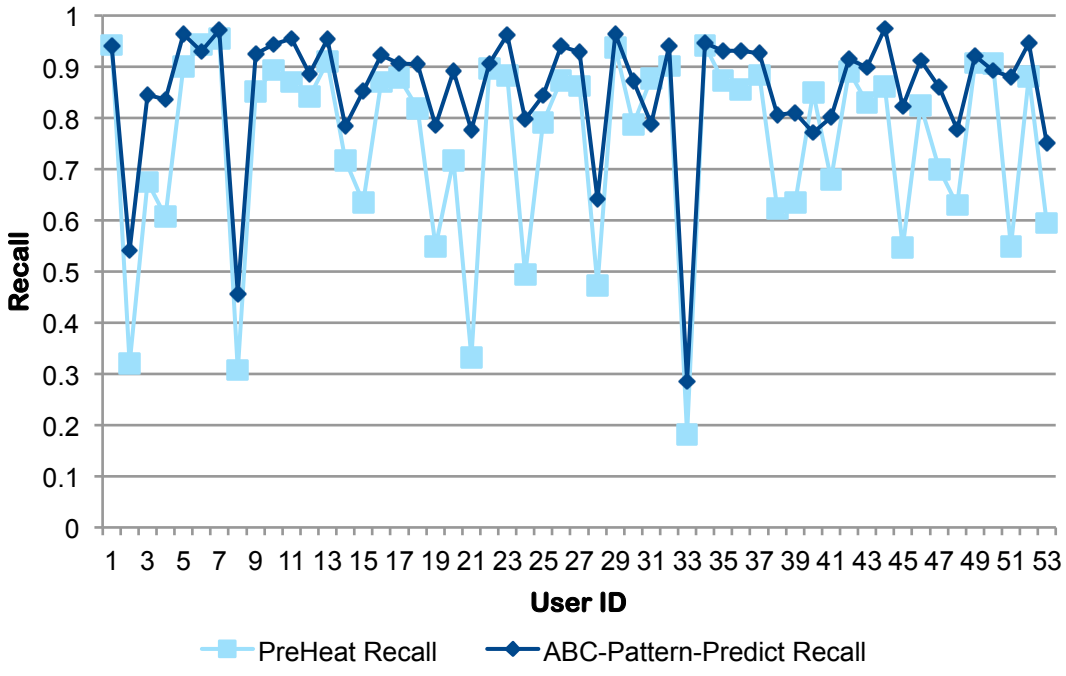

Figure 22 Recall Work-Time 8AM - 6PM

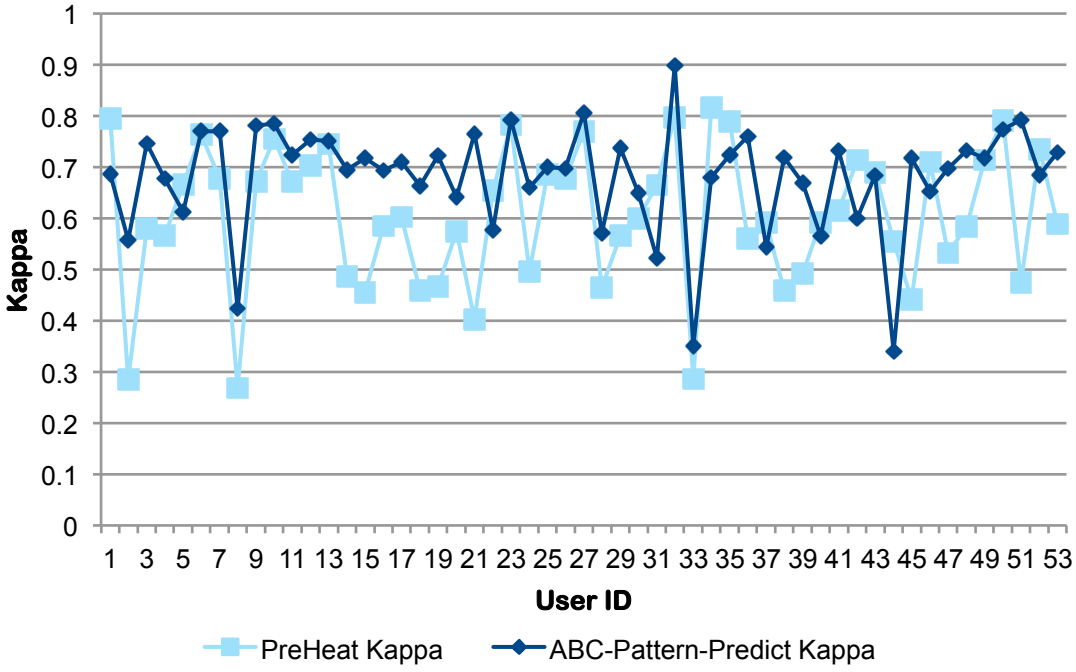

Figure 23 Kappa Work-Time 8AM - 6PM

Figure 16 shows that the All-Day accuracy of ABC-Pattern-Predict is slightly better than PreHeat with an average difference across all users of 1.12% (SD=1.64%). This difference becomes more pronounced for the Work-Time accuracy (see Figure 17) for which ABC-Pattern-Predict is on average 2.93% (SD=4.91%) more accurate than PreHeat. Both ABC-Pattern-Predict and PreHeat outperform both PPM and the ZeroR baseline. Since the difference to PPM and ZeroR is so great we chose to investigate the deeper measures of Precision, Recall, and Kappa only for ABC-Pattern-Predict and PreHeat. As we can see in Figure 19 and Figure 21, both the All-Day and Work-Time precision of ABC-Pattern-Predict and PreHeat is nearly identical with an average difference of 0.04% (All-Day) and 0.72% (Work-Time). Most of the differences between the algorithms come from Recall (see Figure 18 and Figure 22), which shows an average difference of 10.93% (All-Day) and 10.11% (Work-Time) in favor of ABC-Pattern-Predict. This is also reflected in the Kappa metric (see Figure 20 and Figure 23), which shows an average difference of 7.28% (All-Day) and 7.65% (Work-Time) in favor of ABC-Pattern-Predict. Both algorithms show a substantial agreement between the predicted and

ground truth data. Comparing these results to the location prediction using only Singular Patterns (Mode 1 of ABC-Pattern-Predict) we see an average All-Day accuracy of 86.8% (SD=5.6%) and Work-Time accuracy of 72.7% (SD=9.2%).

A statistical analysis, using a Wilcox test, of the All-Day accuracy results shows a significant difference between ABC-Pattern-Predict and PPM ($p < 0.05$) as well as ZeroR ($p < 0.05$), but no statistically significant difference between ABC-Pattern-Predict and PreHeat ($p = 0.31$). This changes for the Work-Time results for which we see a statistically significant difference between PPM ($p < 0.05$), ZeroR ($p <0.05$), and PreHeat ($p < 0.05$). Analyzing the precision, recall, and kappa results we see a statistically significant difference between ABC-Pattern-Predict and PreHeat for the All-Day Recall ($p < 0.05$) and Work-Time Recall ($p < 0.05$) results. In addition we can observe a statistically significant difference between ABC-Pattern-Predict and PreHeat for the Kappa All-Day ($p < 0.05$) and Work-Time ($p < 0.05$) results. As expected we cannot observe a statistically significant difference for the Precision All-Day ($p = 0.65$) or Work-Time ($p = 0.65$) results.

The rest of this chapter is dedicated to evaluate the performance of ABC-Pattern-Predict in a select application domain, efficient office temperature control. First we present a simulation framework that highlights why the usage of predictive algorithms for efficient temperature control is worthwhile. In the last part of this chapter we simulate the temperature control using the occupancy traces provided by the analysis we performed. In order to show the impact of ABC-Pattern-Predict we simulated the control through building simulation software, EnergyPlus, using varying degrees of setback.

6.1. Office Temperature Control Through Indoor Location Prediction

According to the U.S. Department of Energy, buildings constitute about 41% of primary energy usage in the U.S., with commercial buildings contributing half of that. Heating, Ventilation, and Cooling (HVAC) systems in commercial buildings account for about 40% of this energy use [70]. While modern HVAC systems make use of Variable Air Volumes (VAV) units for independent control of thermal zones [31], most modern buildings still use static schedules to run HVAC systems, thereby wasting energy when spaces are unoccupied [8,19,20,24,74,75]. Thus, recent work has focused on actuating HVAC systems, based on near real-time occupancy information [8,19], as well as predicted occupancy-based on learned patterns [37,62], with estimated HVAC energy savings ranging from 30-40% in the best case [19,62].

There is no question that occupancy-based HVAC actuation systems are effective for substantially reducing the energy use in commercial buildings. However, there are significant costs involved in the design, installation, and maintenance of the occupancy data collection network. For example, Erickson *et al*. [19] report an expense of $140,000 just for the hardware of a three-floor building, and even simple wireless detection sensors can cost well into the hundreds of thousands of dollars for entire buildings. While wireless sensors can be incrementally deployed, thereby reducing installation costs, in practice it is very difficult to maintain a large-scale wireless sensor network [33]. Building managers could make better choices about such systems and their upkeep if they could systematically explore the design space of cost, complexity and energy usage tradeoffs between candidate occupancy detection/prediction systems for HVAC control.

To address this challenge we created a simulation framework [In submission: 25] that allows building managers to explore the effect of varying different parameters of occupancy prediction algorithms on energy consumption and occupant comfort. Our framework's inputs are: reference occupancy patterns, false positive and false negative rates of the prediction algorithm, prediction look ahead length, prediction error length, temperature setback settings, and building characteristics. *Prediction look ahead length* is defined as how far in advance occupancy predictions occur,

prediction error length is the temporal clustering of consecutive predictions, and *temperature setback settings* represent the minimum and maximum temperatures that the indoor temperature can reach before the HVAC system engages.

We validate the utility of our framework for assessing energy consumption and occupant comfort using a 196-day longitudinal occupancy-temperature dataset that we have collected for 235 offices. We analyze the effects of varying three parameters in a simulated prediction and control system: *false positive rate* (false positives/(false positives + true negatives)), *false negative rate* (false negatives/(false negatives + true positives)), and *temperature setbacks*. We use a Revit building model [6] of a large university office building on our campus and Energy Plus [15] to simulate building HVAC control in response to predicted occupancy and weather data. We compare the relative benefits of different levels of false positive and false negative rates, and temperature setbacks based on energy consumption, as measured by Energy Plus, and occupant comfort, based on a MissTime [46], the average daily number of minutes a room was not at a chosen comfort temperature. This metric indicates on average how many minutes per day an occupant had to endure temperatures that were not their preferred temperatures.

Our results demonstrate three key factors: first, that predictive occupancy systems with wider temperature bounds can measurably outperform reactive control systems; second, that predictive occupancy systems with smaller temperature bounds can outperform static schedules in energy reduction, with comparable or better occupant comfort; third, that the differences between occupancy prediction algorithms that have near state-of-the-art performance, have fairly large effects on energy consumption and occupant comfort, demonstrating that even moderate gains in performance can be meaningful in a cost-benefit analysis. These results validate the value of our framework, in a number of ways, as it demonstrates the potential utility for building managers in 1) deciding whether to switch from reactive occupancy to predictive occupancy, 2) deciding whether to install occupancy detection technology and choosing between multiple alternatives, 3) deciding whether to upgrade the prediction occupancy system and algorithms currently in

use for the building. In any of these cases, our framework would allow a building manager to make a more informed, and likely better, decision.

6.1.1. Related Work

The significant energy consumption of HVAC systems, coupled with a growing worldwide awareness of sustainability, has led to significant research work on methods to reduce their energy usage. Prior work has shown that HVAC control systems can use occupancy data to optimize HVAC scheduling [8,19,24,47,62]. For example, Erickson *et al.* created an occupancy-based HVAC control system that has the potential to reduce energy consumption by 30% compared to an occupancy oblivious system, while maintaining thermal comfort [19]. Numerous other predictive techniques achieve similar performance [8,24,62].

As mentioned previously, one of the key facets of occupancy-based HVAC scheduling is balancing energy consumption/cost with occupant "thermal comfort" [12,18,19,20,74]. Keeping an HVAC system fully on always would ensure thermal comfort at all times, but significantly increase energy consumption. Similarly, turning off the HVAC system, or running it at minimal levels to keep the building infrastructure intact (*i.e.*, prevent freezing pipes or overheating of equipment) would achieve a much lower energy usage, but lead to significant occupant discomfort. Neither of these extremes, perfect comfort nor perfect energy usage, is desirable, which has led to a number of approaches for balancing these factors [8,18,19,20,24,48,62,74].

One of the most promising approaches, reactive control in which a room is adjusted to its preferred temperature only when an occupant is detected, often relies on costly, *per*-office sensors to track occupancy. However, Balaji *et al.* were able to reduce costs by using existing WiFi infrastructure to track participants' smartphones, yielding a 17.8% reduction in HVAC electrical energy consumption [8]. This approach required users to have a smartphone, be willing to install their app, and did not work in public spaces (individuals' phones were associated with their offices). This system moderately reduced thermal comfort during the time taken by the HVAC system to reach the desired temperature of the occupant.

In contrast to reactive occupancy systems, there is potential for modeling occupancy to create a 'predictive occupancy' system [19,36,62]. Scott *et al.'s* work, on the Preheat system for residential homes, utilized predictive occupancy to reduce the amount of 'missed time', time when the house was occupied but not warm, by 6-12 times compared to a static schedule while using an equivalent amount of energy [62]. Current research has focused on achieving or determining the highest possible detection and prediction accuracy. This is an important goal to strive for; however, it is also important to remember that achieving this goal has costs. Occupancy detection infrastructure, its installation, and its maintenance can be very expensive, particularly when renovating older buildings, rather than new construction [5]. Worse yet, the better the sensors and smarter the HVAC control systems, the higher the cost generally becomes [74]. Building managers need better information to tradeoff the gains in energy efficiency due to increased accuracy against costly infrastructure.

To this end, Kleiminger compares the results of multiple existing occupancy prediction algorithms with differing levels of accuracy, evaluating them both on energy consumption and missed time [36]. However, comparing multiple accuracy levels was a by-product of their exploration of algorithms, not its focus. Balaji *et al.'s* work on the Sentinel system was partially motivated to utilize existing WiFi infrastructure for occupancy detection, and to reduce installation and running costs [8]. Erickson *et al.'s* work on the POEM system includes a return on investment analysis to show its cost-effectiveness [19]. However, neither paper attempts to study the inherent tradeoff between occupancy detection and prediction cost and the benefits of smart HVAC control in a systematic way.

In contrast to prior work, we present a comprehensive framework that supports the systematic comparison of occupancy prediction algorithmic performance using several parameters such as accuracy level, error distribution, and building characteristics. Using our framework, building owners and managers can more accurately evaluate the benefits of installing various occupancy sensing technologies. This allows them to make better decisions with regards to the aforementioned tradeoff between these benefits and the cost of

installation/renovation. To our knowledge no framework exists to allow building managers to explore the tradeoff for differing levels of predictive algorithmic performance.

6.1.2. Simulation Framework

Our primary contribution is a general framework that can be used to systematically evaluate the impact of different predictive algorithm parameters on energy consumption and occupant comfort. As shown in Figure 1, our framework uses a multi-step process to simulate the impact of occupancy prediction errors. The framework performs three steps: (1) prediction of room occupancy, (2) building simulation based on predicted occupancy, and (3) error and impact calculations.

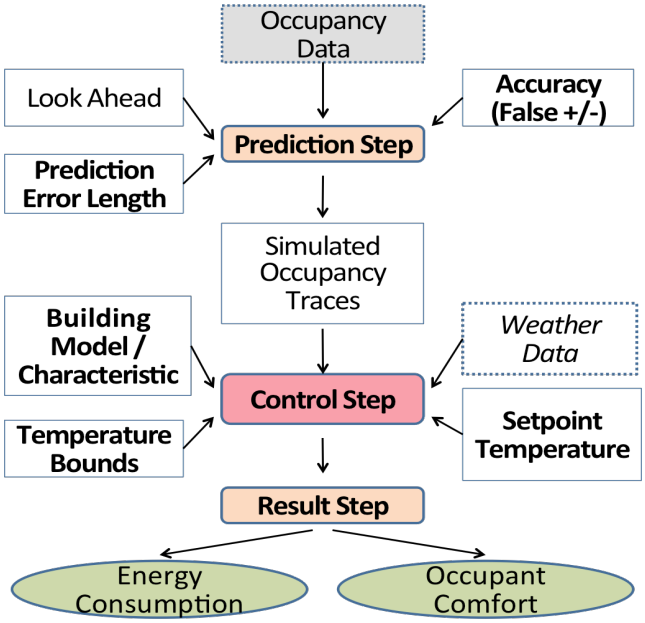

Figure 24. 3 Stages of our Simulation Framework

(Dotted boxes: Inputs; Solid boxes: parameters;)

At each time step, for each room, the framework determines from the data whether that room will be occupied at a look-ahead time specified as part of the simulation (the look-ahead length). Having determined what an Oracle (perfect predictive system), would do, we then determine the likelihood that a correct prediction will be returned given the provided false positive and false negative rates (see Prediction Step in Figure 1). Importantly, while we ensure that the simulation of occupancy prediction for a given room has the overall provided false positive and false

negative rates; we vary these rates throughout the day. It is a well-known fact in building management that occupancy prediction is easier during certain times of day than others. To simulate this effect, we use a weighting function based on the maximum likelihood estimate for a room being occupied or unoccupied at each time of day over the length of our dataset. For example, we use the likelihood of a room being occupied at 3pm each day over the course of the dataset as the weight for the 3pm timeslot. (Note, we calculate weights for each time slot on weekdays and weekends separately, as times that are likely to be occupied during the week are not necessarily likely to be occupied over the weekend).

We redistribute errors such that times during which predictions would be more difficult will have higher error rates. In other words, times of the day that are unlikely to be occupied during our data set will be more likely to have prediction errors occur when they are occupied, as opposed to times of day which are generally occupied, which will be less likely to have errors. Based on this calculated probability that a correct prediction will be returned at a given time, we randomly determine whether an error would occur, and then return either the correct prediction, or an error.

Importantly, errors only affect predictive occupancy. Once the room is actually occupied, we assume that the control system falls back to reactive occupancy, regardless of what the predictive system indicates, and is assumed to correctly monitor the occupant until the occupant leaves the room.

We then use a Revit model of a typical university office building, and create a mapping from our predicted occupancy traces to the rooms in the Revit model. This model is for a real building on our campus and was provided to us by the building managers themselves. Then, we run Energy Plus simulations based on the occupancy predictions, building model, input parameters, and weather data from the target city during the period of analysis. However, any building modeling technique that relies upon occupancy traces could be used.

In the final step, we analyze the results. Our results are split into two categories: energy impact, and occupant comfort. Energy impact is based on energy used by

the HVAC system over the course of our Energy Plus simulation. We measure occupant discomfort with MissTime (see Supported Metrics).

We implemented our simulation framework in Python, with an SQL back end to store data related to the simulations, such as predicted occupancy traces at each time step, as well as the differing parameters used for the simulation. As stated above, we use Revit building models and Energy Plus software for modeling HVAC control and energy usage of the building. The simulation framework's capabilities are described in the next section.

6.1.3. Variations Supported by the Framework

Our simulation framework is designed to support specification and variation along multiple dimensions:

Accuracy; False Positive and Negative Rate

The framework can vary accuracy at any level; however, accuracies below 50% are unimportant for analysis, given that the least accurate algorithms in the current literature give significantly higher accuracies [8,20,19,36]. We define accuracy as the chance that if the model were to make a prediction, it will correctly predict occupancy at that level of accuracy. For example, if the correct prediction is a room being 'unoccupied', a model with 70% accuracy would have a 70% chance of predicting 'unoccupied', and a 30% chance of predicting 'occupied'.

Although varying accuracy systematically is valuable, it lacks nuance. It is quite common for machine learning models generally, and predictive occupancy models specifically; to have differing false positive and false negative rates. Additionally, algorithms can often be tweaked to decrease only one of these error rates, while potentially increasing the other. In light of this, a building manager may wish to examine the effects of differing false positive and false negative error rates. This affords a building manager a greater degree of control of the tradeoff analysis by examining the effects of both forms of error rate independently. Importantly, rooms with differing levels of occupancy will have different overall accuracies at false positive and false negative error rates. A room that is rarely occupied will have fewer total numbers of errors with a low false positive rate and high false negative

rate, than a room that is commonly occupied. To account for this, our framework returns the overall accuracy for the inputted rooms to the building manager. This allows the building manager to examine his results either at the false positive/false negative level, or at an overall accuracy level, which can be particularly important for predictive occupancy solutions that only report overall accuracies.

Look Ahead

A key factor for predictive occupancy control of HVAC systems is how far in advance predictions about occupancy occur. The length of the look-ahead depends both on the temperature change rate of a room as well as the intended deviation from the room's preferred temperature. Too short of a look-ahead may mean not being able to bring the room up to optimal temperatures prior to predicted occupancy, leading to occupant discomfort. Look-aheads that are too long may mean bringing the room up to optimal temperatures somewhat earlier than necessary, leading to HVAC inefficiency. The framework supports a range of potential look-ahead values starting with ten minutes and increasing at five-minute intervals to any value desired. The five-minute interval increase is based on our five-minute data granularity. Changing the interval increase to accommodate more or less fine-grained data is a trivial extension.

Prediction Error Length

Prediction models that use temporal features, which almost all do in the field of HVAC control, will generally have 'clustered' predictions. In other words, a model that will predict 'unoccupied' at time *t*, is more likely to also predict 'unoccupied' at time *t+1* and time *t+2*. This is important, because a model that predicts occupied five times and then unoccupied five times, versus one that alternates between predicting occupied and unoccupied, will have differing performance characteristics. To account for this phenomenon, the simulated model includes a clustering factor when making predictions. This can take one of two forms in our framework. First, there can be a fixed prediction error length, which simply means that instead of determining whether an error will occur at each time step, the decision is made for the fixed length of time input, which can constitute multiple time steps. Second there can be a variable prediction error length. In this case the framework will

randomly choose whether the simulation will continue making that prediction for a time period ranging from one time step to however many time-steps the user wishes to input. While this clustering is an approximation, it is designed to better simulate the patterns of an actual predictive occupancy model using temporal features.

It is important to note that if the model predicts unoccupied for a time length *t* and the room becomes occupied within this length *t*, our framework will shorten *t* to account for it. For example, if *t* is 20 minutes and the room becomes occupied within 15 minutes, then *t* will be shortened to 15. This is in keeping with the earlier stipulation that the system will fall back to reactive occupancy once the room is occupied.

Temperature Bounds

Modern HVAC systems, at a minimum, are designed to maintain the temperature in a building to levels that prevent damage to the building infrastructure. For example, the temperature in a building is kept well above freezing to prevent water pipes from bursting. Most buildings further restrict this range, in part to ensure occupant comfort. However, restricting the range of acceptable temperatures for an ***unoccupied*** room results in wasted costs by making the HVAC system run unnecessarily to keep the room within those temperature bounds. Thus, when using our framework, it is important to specify a minimum and maximum allowable temperature. These minimum and maximum temperatures can vary depending on the room, to take into account differing heating and cooling preferences for different spaces. It should be noted that while room specific temperature bounds may increase accuracy, they also increase the complexity and difficulty of using the framework, and so our framework provides a uniform set of temperature bounds across all rooms, by default.

Building Model

The characteristics of the building and its HVAC system influence the HVAC system's heating and cooling rate, and the rate at which indoor temperature normalizes (based on building insulation) with the outdoor temperature when the HVAC system is not running. This modeling requires information about the outdoor

temperature and cloud cover, since the effect of solar energy on a building's internal temperature can be significant.

Perfectly modeling HVAC heating and cooling rates, as well as temperature normalization or decay rate, is a difficult problem. As a gold standard, we use Revit buildings models with Energy Plus simulations to model energy consumption [6,15].

Importantly, for those who choose not to use a Revit building model, there are other potential options. Temperature change rates could be empirically approximated based on temperature data for each room from the existing building management system. Another option is empirically model the buildings based on collected sensor and actuator data. While these techniques would yield only rough approximations for temperature, they could be reasonable to use for within-building comparisons of the effects of varying other variables. The potential effects on the quality of the output are discussed in the next section.

A salient point to keep in mind regarding building modeling is that the base occupancy traces provided, and therefore the simulated occupancy predictions from the prediction step, need not come from the building being modeled. Indeed, this is quite likely to be impossible in the case of new building construction or renovation of buildings without occupancy detection infrastructure. However, in these cases it is possible to use representative occupancy traces, at least on an aggregate level. This would involve taking occupancy traces from a building with similar usage (commercial office building, university building, campus office building) and size which already has occupancy detection infrastructure, and applying them to the new building. Since the generation of the simulated occupancy traces and the control of the building are entirely separate stages, the fact that the base occupancy traces came from a separate building would have no major effect on the results, assuming that the occupancy traces in question are at least reasonably representative of the target building's usage modalities.

6.1.4. Supported Metrics

We evaluate the results of our simulations in two ways: Energy Impact and Occupant Discomfort.

Energy Impact

As mentioned previously, we use a Revit model and Energy Plus to simulate a building's HVAC control in response to our simulated occupancy predictions. Energy Plus calculates HVAC energy usage on a monthly basis at a room-by-room level, and we use this calculation as our Energy Impact metric. We aggregate the room-by-room and monthly data to analyze at the building level over the entire simulation period for each simulation.

	Predictive Occupancy	Reactive Baseline	Static Baseline
Temperature deviation bounded by 2°C (~4°F) setback	Run	Run	
Temperature deviation bounded by 6°C (~11°F) setback	Run		
Temperature Deviation bounded by 10°C (~20°F) setback	Run		Run

Table 13 Simulation Conditions

As stated above, building managers can choose a custom energy calculation that does not depend on a building model-based technique. In this case, we recommend an equation provided by the U.S. Department of Energy (DoE) [68], which has been used in the research literature [7]. This idealized equation takes as input the difference between room and outdoor temperature, as well as locale specific constants, and returns estimated energy consumption in kWh. While inherently less accurate than a building model simulation, this equation is generally applicable and available to all building managers.

Occupant Discomfort

Our framework supports an estimate of occupant discomfort by analyzing MissTime [46], the average daily number of minutes a room was not at the correct comfort temperature. This metric indicates on average how many minutes per day an occupant had to endure temperatures different from her preferred temperature. We differ from Preheat, an accurate occupancy prediction algorithm that used MissTime as a measure of occupant discomfort [46] insofar as we assume a fixed preferred comfort temperature and not a comfort temperature range. This limitation

is necessary since Energy Plus does not give us the ability to export the daily room temperature distribution in a 5-minute interval and thus we cannot calculate when a room reaches the comfort temperature range. This will overestimate the comfort impact, but we believe it will still provide a valuable validation point for our framework.

6.1.5. Framework Validation

The goal of our validation is to show specific differences in the performance of simulations varying algorithm parameters that building mangers could use to make better decisions as to what type of occupancy prediction hardware and software that will be most appropriate for their needs.

Unfortunately, the Revit model for the building that we used to collect occupancy traces was not available to us. Instead, for validation we used the Revit building model for another representative building and created a mapping between the two buildings. While not ideal, this does not significantly reduce the validity of our results. As mentioned previously, the simulation of occupancy prediction and the control of the building are entirely separate stages of our framework. Furthermore both buildings are large university buildings, suggesting that the occupancy traces for one building should be reasonably representative of the other. Additionally, as stated above, the purpose of this validation is to show that varying the factors we have chosen to analyze has a measurable effect on a building's energy consumption and occupant comfort. We do not seek to show that this is true for a specific building, rather to provide a proof of concept that this framework yields results that can be useful for real buildings generally. Since our Revit model accurately depicts an existing building, its results will serve just as well to validate our framework.

Our data set includes 196 days of occupancy data collected at five-minute intervals, from June 2011 to December 2011 for the GHC building. This data included the room ID, a timestamp for when the data was collected, an outdoor temperature, an indoor temperature, and a binary occupancy status. Occupancy was measured using dual-technology (Passive Infrared and Ultrasonic) sensors deployed across

all rooms in our testbed building, networked to the existing Building Management System.

Currently, the building uses a reactive occupancy system. We use this control system as a baseline to compare our simulations against. Additionally, we simulated a static 6am-9pm schedule system, as another baseline to compare against.

Simulation Parameters

Our simulation framework has a wide range of inputs and parameters. While varying all of these parameters to their fullest extent would be ideal, due to space constraints we focus on parameter values that are the most realistic, and will have the greatest effect on our metrics. In particular, we vary the false positive and false negative error rates, and the temperature bounds in our validation, for a total of 27 simulations. Additionally, we simulate a reactive system and a static schedule system as baselines to compare our results against.

False Positives/Negatives: Many variations

The main goal of our validation was to demonstrate the benefits of simulating different levels of false positive and false negative rates, and compare their impact given other parameterizations of the framework. We chose to focus our analysis on false positive and false negative rates as most designs of predictive occupancy algorithms and control system report their performance either in terms of overall accuracy or false positive and false negative rates. Therefore, we argue that understanding the gains from differing levels of false positive and false negative rates will provide building managers with a greater degree of insight into the value of differing systems. Our experiments were conducted at 25%, 15%, and 5% false positive rates crossed with 25%, 15%, and 5% false negative rates (9 total error rate variances). We chose these values as representative of the range of performance for ‘state of the art’ prediction algorithms.

Look Ahead: Fixed Value

As stated earlier, look ahead should be defined based on the time necessary to bring a room to a comfortable temperature. Due to space constraints, we only report results for one Look-Ahead value. We choose 60 minutes, both as a

reasonable time frame to perform predictions over, as well as matching with our longest possible prediction error length (see blow).

Prediction Error Length: One Value

We configured the *prediction step* of the framework to use *Random-Prediction* errors length, last randomly from five minutes (the shortest time-step in our dataset) to one hour. These random prediction errors had a mean length of 32.5 minutes (SD=13.8 minutes).

Temperature Bounds: Three Variations

Setting temperature bounds relies on two variables: the heating and cooling setpoints when a room is occupied, and the length of the bounds between occupied and unoccupied setpoints. We chose 20°C (68° F) as our occupied heating setpoint, and 24°C (75°F) as our occupied cooling setpoint. We chose these setpoints based on the ISO 7730 standards of comfort [34]. We also simulated three different temperature bounds, as described in Table 1: The first

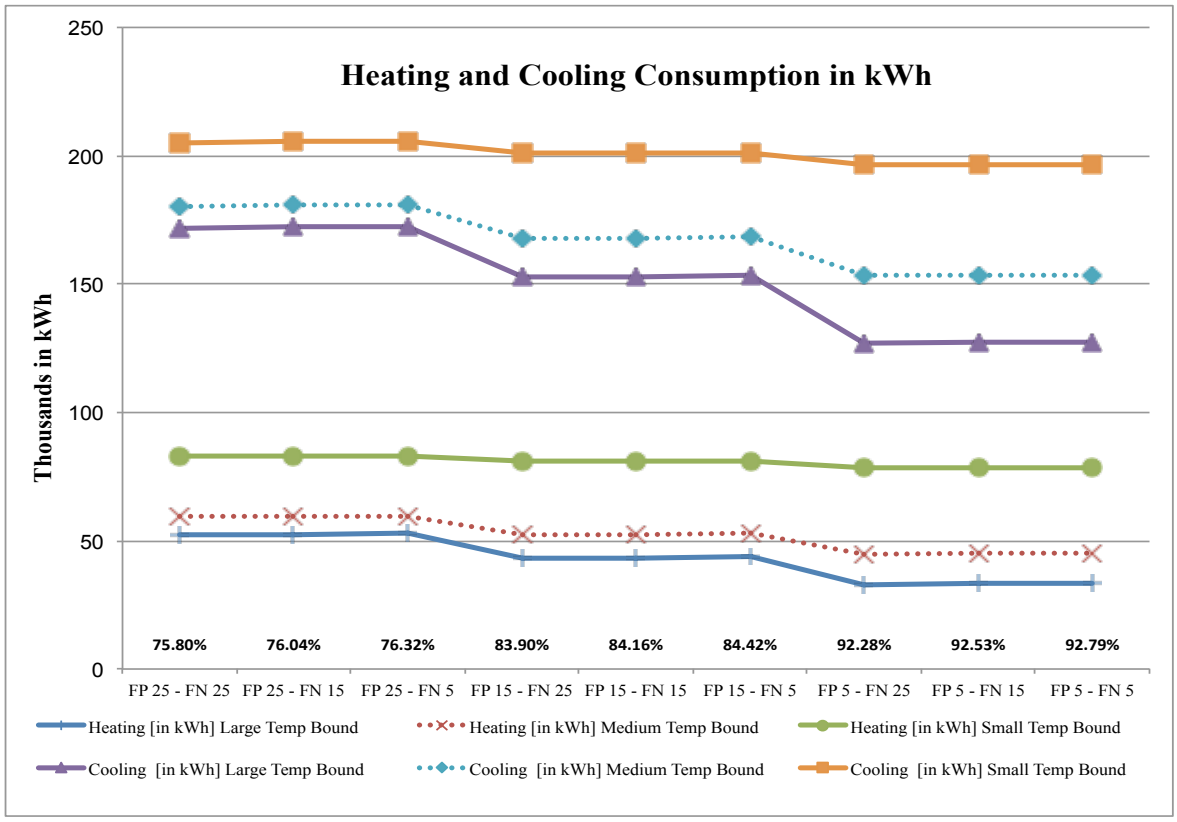

Figure 25 Overview showing energy consumption, in kWh, of the 27 EnergyPlus predictive occupancy simulations run over six month. % for each data point is average accuracy across all rooms.

bounds (*Small-Bound* in Table 1) are 2°C from the setpoint temperatures. We chose these as the most rigid realistic bounds, similar to those used in reactive control systems. The second set of bounds (*Medium-Bound* in Table 1) are 6°C from the setpoint temperatures. We chose these as they were well within safe operating parameters for the building, but would provide some notable energy savings, without jeopardizing occupant comfort in most cases. Finally, the last set of bounds (*Large-Bounds* in Table 1) is 12°C from the setpoint temperatures. These represent the most extreme temperature bounds that a building manager could use, as allowing the temperature to float further could have negative effects on building operation. While unlikely to be put into practice, this final condition is used to show the upper bound on energy gains that a building manager could achieve. Additionally, we run the reactive system with small-bounds, and the static schedule with the large-bounds case as comparisons.

Building Modeling:

As stated earlier, we made use of a Revit model of a large university building on our campus, and used Energy Plus to simulate the effects of HVAC control and weather for the 6-month period between June and December in response to our simulated predicted occupancy traces. We choose a multi-seasonal time period to validate our framework over a range of weather periods as well as increase the sample size of our occupancy traces.

It is important to note that we did not simulate every HVAC zone in our Revit model, only those for which we had simulated occupancy traces to map to. This has little bearing on comparisons between conditions; however, overall aggregate energy usage as well as monetary savings, are only for the 235 offices we simulate, not the entire building.

Evaluation Metrics

To quantify the energy impact of different options we use the output of our Energy Plus simulations. For occupant comfort, we use the aforementioned variant of MissTime. We compare in terms of both absolute, and percentage reductions.

Evaluation Results

For each simulation scenario we generated, we calculated two metrics (totaled over the 6 month period of our occupancy traces) – Energy Impact and Occupant Comfort. We compare the predictive occupancy simulations to themselves, to the reactive baseline, and to the static schedule baseline.

6.1.6. Energy Impact

Figure 2 shows the heating and cooling HVAC energy usage for the offices we simulate in our test building under each of the predictive occupancy simulation conditions in kWh.

As expected, wider temperature bounds reduced energy impact in each case. While unsurprising on a theoretical level, it is important to note that rather than having a theoretical understanding that wider temperature bounds reduce energy usage, these results let our building manager know specifically that widening from the small to medium bounds led to an average overall savings of 148,602.46 (kWh), or 24%, and between the medium temperature bounds and large temperature bounds and average overall savings of 124,379.84 (kWh) or 28%. This is particularly important as the relationship between widening temperature bound (by number of degrees) and energy reduction is obviously non-linear, and thus not easily calculated without testing specific temperature bounds.

The second important result in Figure 2 is also one that is theoretically expected. Decreasing the simulated false positive rate led to declines in energy usage under all temperature bounds. However, once again the value of our framework comes not from answering the theoretical question, but from quantifying it in terms that a building manager can use. A building manager using our framework would know that a decrease from 25% to 15% false positive rate showed an average 18,272 kWh reduction in energy impact, or 7.27% total power usage, and similarly that a decrease from 15% to 5% false positive rate showed an average 21,824 kWh reduction in energy impact, or 9.30% total power usage. This is particularly useful piece of information because the relationship between false positive rate and energy impact is non-linear, making the increased number of datapoints our framework provides even more valuable.

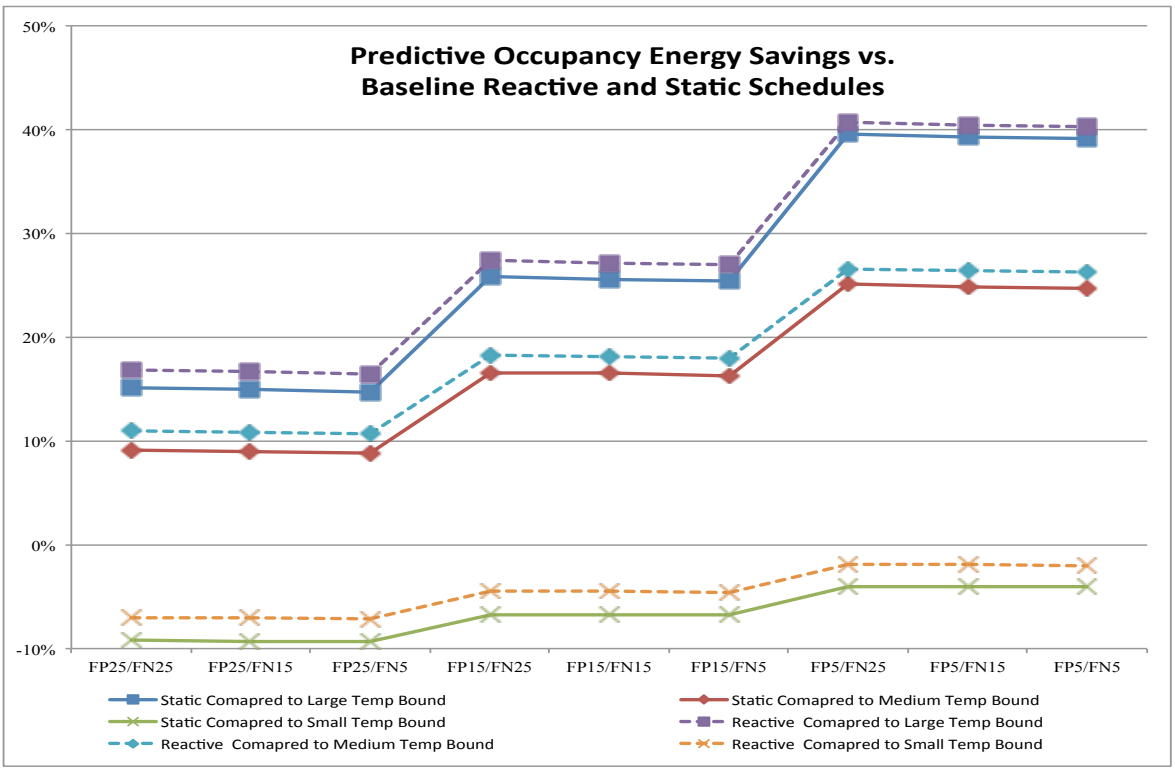

Figure 26 Energy Savings Predictive Occupancy systems vs. Baseline Reactive and Static Schedules

Percent Savings Compared to Baselines

Figure 3 shows the relationship between overall energy consumed by each of the 9 simulated occupancies, and the reactive and static baselines established. As expected, the reactive system outperforms all levels of predictive occupancy using the same small temperature bound. However, it is important to note that the medium and large bound cases both outperform the reactive system. More importantly, our framework quantifies how much these larger bound cases outperformed the reactive system. Depending on false positive and false negative rate, our building manager could expect to see a 10.68% to a 26.62% reduction in energy usage by changing from a reactive system with small temperature bounds to a predictive occupancy system with medium temperature bounds. Similarly, the conversion to a large temperature bound system would yield a 16.44% to 40.73% reduction in energy usage.

The static 6am-9pm schedule with large temperature bound outperformed all predictive occupancy cases with small temperature bounds. However, the occupancy cases with large and medium temperature bounds both uniformly outperformed the static system. This is somewhat unsurprising in the case of the large temperature bounds, since most occupancy occurs during the day and therefore the static system will be 'on' more often than the predictive system. However, it is important to note that our framework provides building managers with quantified values for these gains, from 14.7% to 39.5%. The fact that the predictive occupancy cases with medium temperature bounds also outperform the static schedule is more surprising, since the effect of temperature bounds is generally quite large, and would therefore not necessarily be intuitive to a building manger. In addition, our framework quantifies the amount that the predictive occupancy algorithms with medium setbacks outperform the static system by, showing an energy impact decrease from 8.8% to 25%.

Occupant Discomfort

We also report the results of our occupant discomfort metric, as can be seen in Figure 4. As expected, the lower false negative rates had significant impacts on the average daily MissTime. However, once again our framework allows us to quantify this effect for our building manager: the average difference between the 25% false negative rate and 15% false negative rate was 5.3 minutes a day, or 13.5%, and the average difference between the 15% false negative rate and 5% false negative rate was 5.1 minutes a day, or 15.6%. While comfort does not lend itself as well to a direct price comparison with cost, the quantified difference in occupant comfort that different predictive algorithms would achieve has obvious utility for building managers choosing between algorithms.

Additionally, unsurprisingly the predictive occupancy systems outperformed the reactive system in terms average daily MissTime. The reactive system in fact had one of the worst MissTime at 213 average daily minutes; however, this may be due to the strict standard of comfort that we describe above. Nevertheless, our validation provides our building manager with quantified values of this reduction: 170-185 average MissTime minutes a day less in predictive occupancy cases.

Finally, the predictive occupancy systems also outperform the static schedule in terms of occupant comfort. This result is somewhat surprising, since the static schedule has perfect occupant comfort from 6am-9pm. Nevertheless, as a result of it has 124 average missed time minutes, suggesting that on average there is about 2 hours of occupancy a day outside of this time band. This is between 80-95 average more MissTime minutes than the predictive occupancy systems.

It is important to note that these results may be specific to university occupancy patterns, with students and faculty working odd hours. While they are important for our building manager, a building manager for an industry office building would need to use occupancy traces either from their building, or from a similar industry

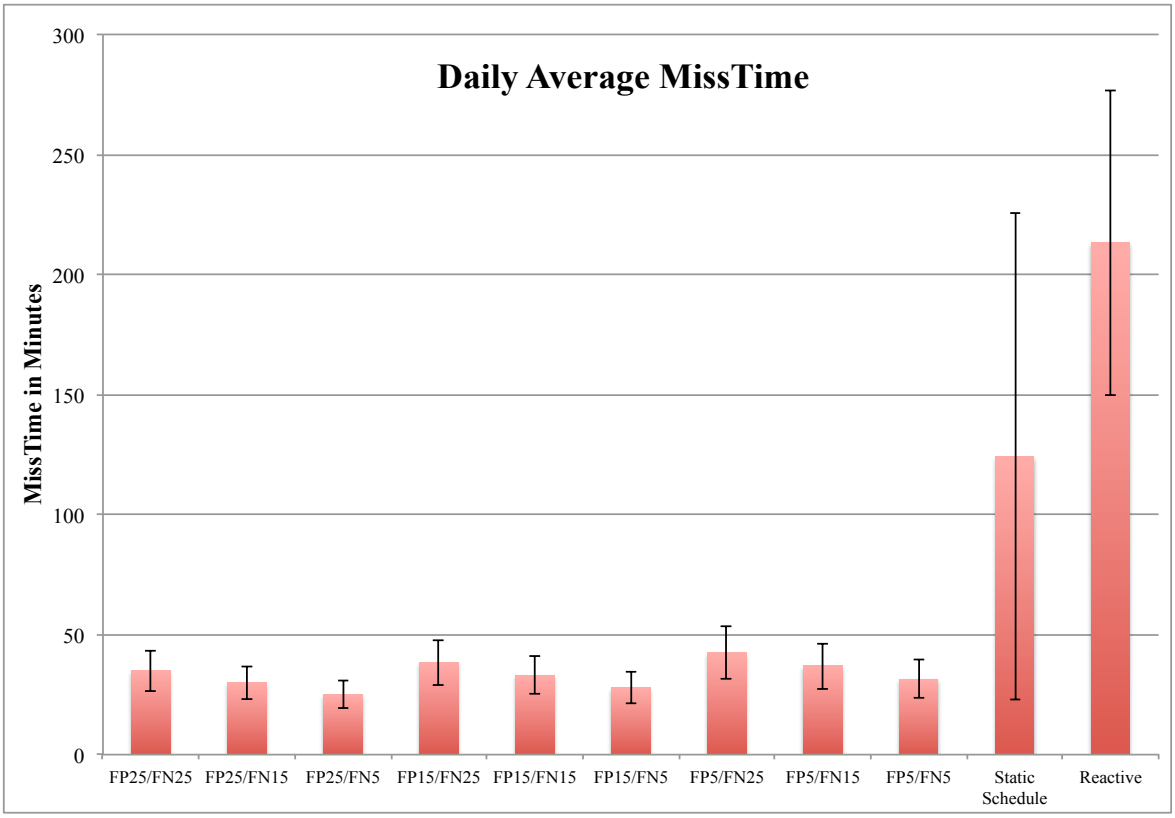

Figure 27 Average Daily Occupant Discomfort across predictive occupancy conditions and baselines.

office building, to achieve accurate results.

6.1.7. Discussion

The results from the validation of our framework are interesting; however, it is important to put them in context. The results presented above are not universal to

all occupancy prediction algorithms with the parameters we chose. Rather, our results are specific to the occupancy traces, building model, and weather data we used in the validation of our framework. In addition, while Revit models and Energy Plus simulations represent the gold standard for building modeling, any model has limitations. In particular, determining the errors bounds of Energy Plus simulations, to provide building managers with a quantified value of the potential inaccuracies of our results, remains a matter for future work. While the effect of such error bounds will be mitigated by the fact that we perform within-building comparisons, they may have some impact on the results.

This does not limit the importance of our validation. Rather than demonstrating that certain algorithm parameters are better for all buildings, climates, and occupancy traces, the importance of our framework is in demonstrating the types and scope of information that would be available to a building manager using our framework, and how this information would allow a building manager to make more cost-efficient decisions and tradeoffs. We couch this discussion in terms of the results from our validation to provide real-world cases of information that would be beneficial for building managers in their decision making process.

Reactive vs. Predictive Occupancy

As a point of common knowledge supported by our validation, a reactive occupancy control system outperforms predictive occupancy control systems with regards to energy consumption when the same bounds are used. However, as stated at the beginning of the paper, to ensure occupant comfort, reactive systems only use small bounds, whereas predictive control systems can use much larger bounds. Of particular importance to our validation was that the medium temperature bound predictive occupancy systems outperformed the reactive system in terms of energy consumption, while maintaining occupant comfort. These predictive occupancy cases consumed approximately 20.8% less energy than the reactive case. From the perspective of a building manager using our framework, if they were currently using a reactive occupancy system, then they could expect to achieve a 20.8% energy savings, depending on the predictive occupancy algorithms they deploy. Depending on the costs of electricity in the area,

as well as the costs of differing predictive occupancy software or hardware, in the case of algorithms requiring more complex sensing technology and feature sets 19 the building manager could confidently choose a software package or hardware upgrade that maximizes his savings, and maintains occupant comfort, with the added benefit of reducing overall energy usage for sustainability purposes.

Static vs. Predictive Occupancy

The discussion above focuses on a building manager who already has occupancy detection technology installed, and is considering the best way to employ or upgrade it. Of equal or possibly greater importance is the case of building managers or owners who do not have occupancy detection technology installed, and have too little information to make informed decisions as to whether to install occupancy detection technology, as well as how to use it.

From our validation, we ran a single simulation with a static control schedule, which kept the temperature to the setpoints between 6am and 9pm, and allowed the temperature to float to the large bounds at all other times. While this schedule performed relatively well on our occupant comfort metric (since very little occupancy occurs outside of these timeframes), it also consumed significantly more energy. Once again, comparing to the medium temperature bounds predictive occupancy simulations, these predictive simulations performed significantly better than the static schedule in terms of energy consumption. Additionally, the results for occupant comfort show that the predictive occupancy systems significantly outperform even a 15-hour static schedule. For a building without occupancy detection technology already installed (new buildings or renovations), this information, coupled with electricity pricing in the area, as well as the cost of installation of a specific system, would give the building manager sufficient information to perform a cost-benefit analysis, and determine whether to install occupancy detection, and to use algorithms that achieve the specified accuracy.

Comparing Predictive Occupancies

- Another important case is considering whether to upgrade a pre-existing predictive occupancy system in a building. As has been mentioned many

times in this paper, the field of predictive occupancy is a vibrant one, and new predictive solutions and systems come out every year. Therefore, it is quite possible that a building manager using one predictive occupancy control system might want to consider a new predictive software solution.

- Our validation shows the potential benefits in such a setting by demonstrating the reduction in energy usage that differing levels of predictive accuracy achieve. A building manager with a control system with a 15% false positive rate could determine, based on the cost of the software and estimated cost of transitioning to a new software package or upgrading their occupancy detection hardware as in 6.1, whether a new algorithm with only a 5% false positive rate was worthwhile to install.

Generalizability

The results from the validation of our framework are based on the occupancy traces and building model we used. However, the analytic framework we developed has significant broader applicability. An analysis of the occupancy for our validation shows a mean occupancy of 20.2% (SD=11.02%). The high standard deviation demonstrates that rooms had varying occupancy rates, and showcases both that current building-wide static schedules are not efficient in such a modern office building, and that the data upon which we base our validation represented a range of occupancy patterns, suggesting that our results can be generally applicable to a range of occupancy patterns. Additionally, we perform our analysis over a 196-day long data set for 235 rooms. While our validation is specific to the buildings we modeled, factors such as weather, building model, and occupancy patterns, are largely representative of other buildings. Further, the framework can be applied to other settings by providing new occupancy traces, building models, weather data and details of the predictive system being considered.

Utility

By providing the design parameters of the building, either through a building model or through approximations of the proposed HVAC system and insulation type, and acquiring a representative sample of occupancy behavior (such as using

occupancy data from a similar building as an approximation), any building manager can determine the levels of predictive accuracy that will provide them with the best cost/benefit tradeoff. Additionally, with these design characteristics in hand, a building manger could also compare the effects of multiple other factors, including those related to the prediction algorithm, such as prediction error length and look-ahead length, and those related to building management, such as the temperature bounds to set, all at once. Also, for building managers who already have detection and predictive technology in place, they may choose to seek out and make use of more or less complex and or expensive control algorithms (*i.e.*, software), based on the benefits that our framework estimates such algorithms would achieve.

Additionally, while we focus our validation on varying control system factors, such as false positive and false negative rates, and temperature bounds, building managers could also use our framework to make decisions for potential design elements for future buildings or renovations. Specifically, they could use the framework to experiment with the effects of installing certain HVAC systems or levels of insulation.

6.2. Energy Consumption Evaluation of ABC-Pattern-Predict

As we were able to see in the results of the simulation framework presented in section 6.1 there is great value in controlling HVAC systems through predictive algorithms. In order to see the impact of ABC-Pattern-Predict on a real-world temperature control system we utilized the same EnergyPlus simulation strategy and evaluated the algorithm under the following conditions: variable setback from 2°C to 14°C in 2°C increments and two different temperature conditions, Pittsburgh and Los Angles. We chose Pittsburgh as a reference as it has variable temperature conditions during winter times (it ranges from relatively mild to very cold conditions), while Los Angles was chosen due to its extremely mild winter. We chose 22°C (24°C Cooling) as the preferred room temperature. We used the same dataset as for the predictive analysis: 53 rooms during Nov. '11 and Dec. '11.

Figure 28 shows the cumulative energy consumption by temperature condition and setback across all rooms. As we expected from our previous simulation the setback has a great impact on the general energy consumption. We can observe

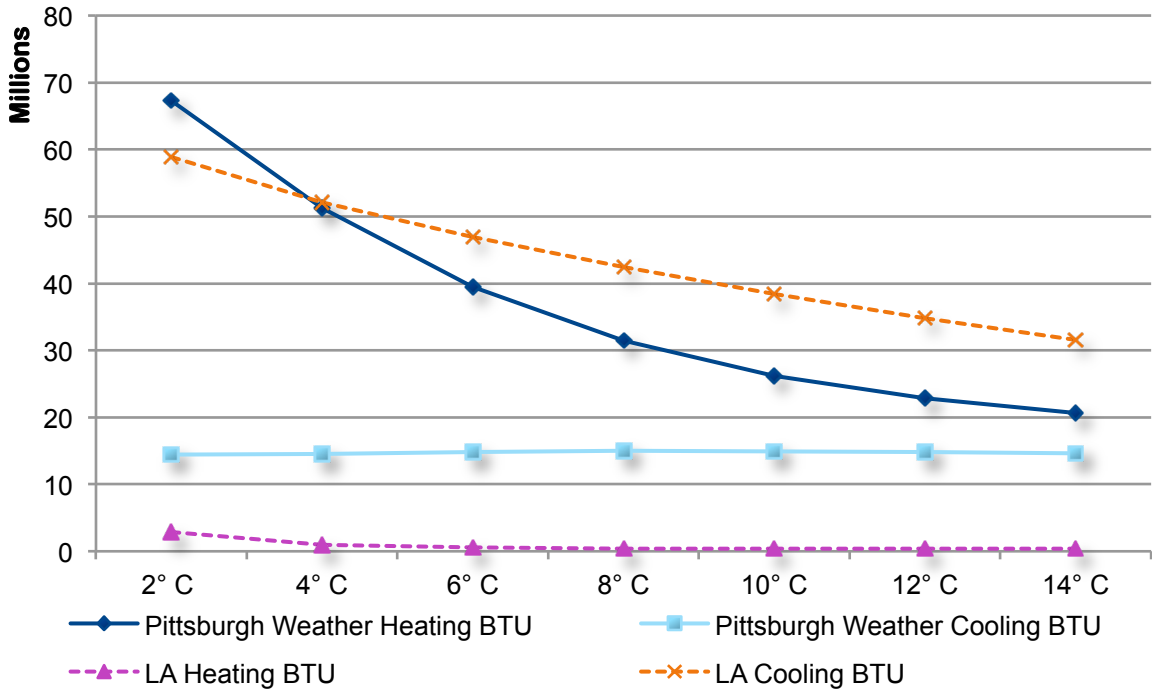

Figure 28 ABC-Pattern-Predict Energy Consumption for Pittsburgh and LA

that the behavior for Pittsburgh and LA are opposite to each other: Pittsburgh has the greatest consumption for heating, but constant cooling and LA shows the opposite behavior. Another difference between these two locations is the relative impact of the setback. The heating consumption shows a much steeper decline for the Pittsburgh condition as the cooling consumption for LA. This shows that during winter month a larger setback might not have a significant impact on the consumption for LA. This situation is most likely reversed during summer month since Pittsburgh shows similar summer temperature as LA winters, but the summer temperatures in LA are much higher. November and December (our evaluation months) is the start of winter in Pittsburgh with average low temperatures of 34.7°F and 25.3°F respectively. Given these low temperature it is not surprising that increasing the setback from 2°C to 4°C is already saving us 23.8% in heating. In Los Angeles, the average high during these two months is only 72.8°F and 67.7°F. This is close to the chosen preferred temperature of 75°F, which means that a higher setback has less of an impact on the overall savings. This is reflected in the more linear and less steep cooling consumption reduction we observe in Figure 28.

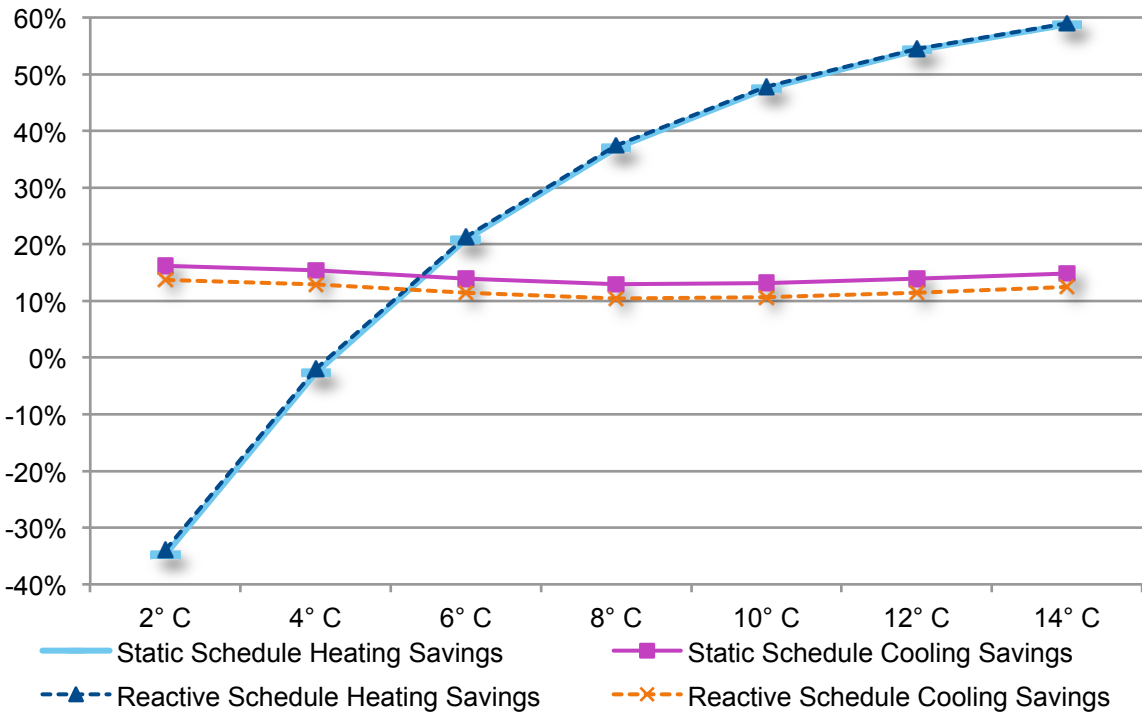

Figure 29 Consumption Savings of ABC-Pattern-Predict over Static Schedule and Reactive Schedule (Pittsburgh Weather)

In order to see how this consumption compares to a reactive and static schedule system, we also computed the consumption savings for the Pittsburgh weather condition by computing the change in consumption from ABC-Pattern-Predict to a static schedule (comfort temperature of 22°C work-time 6AM to 9PM, 12°C otherwise) and a reactive schedule (22°C while room occupied, 20°C otherwise). The results of this analysis can be seen in Figure 29. As we can observe for a 4°C setback (64°C setback temperature) the savings are nearly 0%. A setback temperature of 6°C (61°F setback temperature) or higher will result in savings of ABC-Pattern-Extract. A moderate setback of 8°C already results in nearly 40% savings. It is to note that the heating consumption between the static and reactive schedule is nearly identical, which can be seen in the consumption comparison.

Since our comparative analysis between ABC-Pattern-Predict and PreHeat showed that the main difference between the algorithms comes from an increase in Recall, we also calculated the MissTime results (shown in Figure 30). As we can see the increase in Recall has a profound impact on the average daily MissTime minutes. The average over all user is 19.7 minutes (SD=11.5 minutes) for ABC-

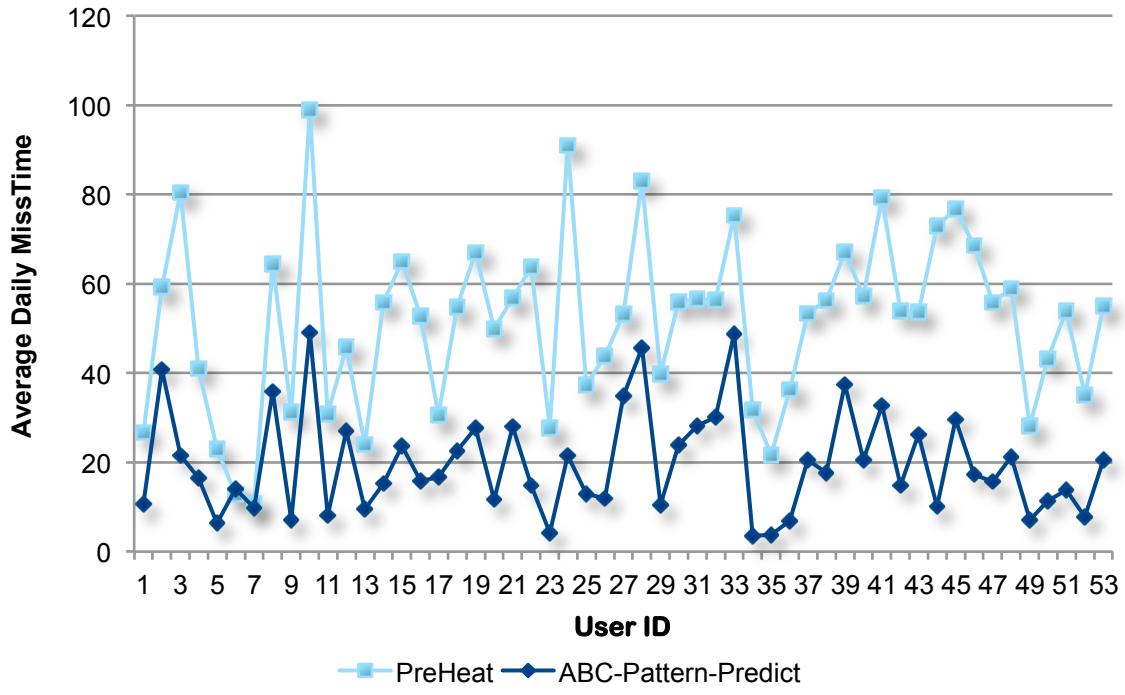

Figure 30 Average Daily MissTime in Minutes

Pattern-Predict and 51.4 minutes (SD=19.5 minutes) for PreHeat, which is an average reduction of 31.7 minutes. A statistical analysis of the results using a Wilcoxon test reveals that there is a significant effect between the groups ($p < 0.05$).

7. Discussion

This chapter conceptualizes the performance of ABC-Pattern-Predict along the metrics of accuracy, precision, recall, kappa, and energy consumption. In addition it highlights areas for improvements and shows how the prediction based on Singular Patterns differs from the full FP-Tree prediction.

Accuracy Performance

The analysis results of ABC-Pattern-Predict show that the algorithm is capable of reaching a very high performance. As we mentioned in the introduction it is very difficult to accurately predict a person's spatiotemporal routine due to the complexity and variations in their daily spatiotemporal structure. The Work-Time ZeroR baseline accuracy shows us how difficult it is to make a decision if a person is at a location or not. On average the ZeroR predictor was only correct in 78.13% (SD=8.67%) of the All-Day cases and 48.19% (SD=18.57%) of the Work-Time cases. Given these low baseline results an accuracy of over 90% indicates a very strong result. A comparison of the algorithm against Prediction-by-Partial-Match (PPM), an algorithm that makes decisions solely based on the time of day, also shows that a person's spatiotemporal structure cannot be captured by only considering one feature. In fact the PPM result is very similar to the baseline accuracy.

Chapter 3 showed that PreHeat is one the best algorithms to predict whether individual offices are occupied or not. Even though simple in its design, the ability of PreHeat to model the spatiotemporal structure allows it to make very accurate predictions about future spatiotemporal events. Even though PreHeat is already showing a very high performance, our algorithm, ABC-Pattern-Predict, can even further improve on the PreHeat results. Even though the differences on an All-Day basis are not statistically significant, a fact that is not surprising given the high All-Day performances, we saw how ABC-Pattern-Predict still achieves large improvements over PreHeat as soon as we limit our analysis to the interesting timeframe of 8AM to 6PM on workdays. As we mentioned in Chapter 3 and 4 it is

necessary to do a more detailed performance analysis in order to understand how the two algorithms differ.

Precision, Recall, and Kappa Performance

Our analysis of ABC-Pattern-Predict and PreHeat showed that the biggest difference comes from an improvement in Recall. This means ABC-Pattern-Predict is making fewer mistakes in predicting that a person is not at a location even though they are (False Negative). Depending on the application domain, this can have a huge impact. In our domain example it will result in less impact on a person's comfort, which will increase the acceptance of the system. On the other hand we saw though that ABC-Pattern-Predict and PreHeat are very similar in precision, which in our domain example results in a higher level of False Positives and unnecessary heating and cooling. As mentioned in Chapter 6, the Kappa results show a substantial agreement between the predicted and ground truth data.

Algorithm Thresholds

Throughout the description of ABC-Pattern-Extract and ABC-Pattern-Predict we used a number of thresholds that were initialized for our experiments, but open up avenues for further research. We used the following thresholds for the extraction algorithm: a pattern frequency threshold that determines when a pattern is extended by another Singular Pattern and the self information threshold that determines when the overall pattern expansion is stopped. The prediction algorithm has two thresholds: for the prediction using only singular patterns we used a 5% probability threshold to retrieve a list of the most likeliest Singular Patterns and for predictions using FP-Trees we used a relative frequency threshold of 0.5 to decide between the results of the MLE and wMLE. Each of these thresholds have implications for the performance of both the extraction and prediction algorithm.

Pattern Frequency Threshold

This threshold has two potential implications: 1) A low threshold will allow ABC-Pattern-Extract to capture more of the found structure, but will increase the computational as well as space complexity, since more pattern are extracted and more probabilities must be calculated; 2) A high threshold will limit the number of

extracted patterns and thus reduce the number of calculated probabilities, but it potentially makes the model less expressive. This threshold can either be determined based on the application domain or empirically from the data, by performing a parameter training using the performance of the prediction algorithm as the objective function. For the latter option enough data needs to be provided.

Self-Information Threshold

The overall pattern extraction is stopped if the self-information of the collection of patterns in one step is monotonically falling over the last three steps. This is primarily a computational complexity criterion that was put in place to limit the overall complexity. It can be either replaced with a more fine grained criterion that makes decision for expansion of individual groups of patterns instead of the whole pattern collection or removed altogether if enough computational resources are available. This criterion is important mainly in the initial training phase, when a new pattern base is created, but it is not used later on in the incremental retraining during prediction run time.

Probability Threshold

This parameter has two possible implications: if it is defined too narrowly it gives too much weight to the absolute pattern probability and might pick a pattern that is not appropriate given the current contextual situation; if the threshold is chosen too broadly it might pick a pattern that seemingly fits the current contextual situation, but does not have enough probability. We picked 5% for our experiments, but it can also be learned from the data by using gradient ascend with the accuracy or precision/recall as objective function. Creating a threshold for individual times of day might also further improve the performance. For example during the night it might be better to pick a more narrow threshold and a more broad threshold during the day. The implications of either should be investigated in future work.

Relative Frequency Threshold

Determining if an event is frequent or not depends highly on an individual and the contextual situation and is an area of active research. For simplification we chose the relative frequency between the MLE and wMLE target to decide between the

two results. Decreasing this threshold means that the MLE result will only be picked if the frequency of the target MLE Singular Pattern is much lower compared to the frequency of the wMLE Singular Pattern. Increasing the threshold would result in the opposite situation. The relative frequency allows us to give more or less weight to the individual results. For example a threshold of 0.5 means that the wMLE target needs to be twice as frequent as the MLE result. Again as before this parameter can be learned from the data. We believe 0.5 is a reasonable threshold, but performance improvements can be gained by adjusting this threshold to individual users and situations.

Potential For Improvement

Even though it has high performance, the results also show that ABC-Pattern-Predict has room for improvement: a few users show better performance using PreHeat. An analysis of the results shows that errors occurred in cases of infrequent, irregular events. PreHeat is capable of reacting to these since it is does not model how likely it is that an event occurs. For example if a person arrives in their office at 6AM in the morning three times in the training dataset, then PreHeat will be able to react to this event if it occurs in the validation dataset. It makes no judgment about how likely it is that this event occurs. We designed ABC-Pattern-Predict to react to two types of events: frequent and infrequent regular events. We defined regular events as events that occur at least four times in the training dataset. This was done both to focus the algorithm on the abovementioned events as well as to reduce the computational modeling complexity. In future work we plan to overcome this shortcoming by allowing our algorithm to also react to irregular events. It should be noted that even though our sample only had 10 student offices (each of them has more than two occupants), PreHeat performed more accurately for seven out of the ten offices. Student offices are the most difficult to predict offices since their schedule is a combination of multiple schedules. Modeling the schedules of each individual will most likely alleviate the situation and allow us to increase the performance for these offices.

If we compare the accuracy differences between ABC-Pattern-Predict and PreHeat (limited to Work-Time) by occupation, we see that we have 23 staff offices in the

sample with an average accuracy difference of 3.22% (SD=4.66%), 14 faculty offices with an average accuracy difference of 5.18% (SD=5.67%), 10 student offices with an average accuracy difference of -0.77% (SD=3.57%), and six offices with unknown occupation with an average accuracy difference of 2.75% (SD=2.51%).

Singular Pattern Based Performance

In addition to the results of ABC-Pattern-Predict Mode 2 (predictions made based on the full FP-Trees), we were also interested to see how successful the prediction is for Mode 1 (predictions based on Singular Patterns alone). Since Mode 1 is much easier to train and test it might be worthwhile in certain situations to use that variation over the whole FP-Tree approach. This is especially true for users for which the majority of the spatiotemporal structure is encapsulated in Singular Patterns. Our analysis shows an average All-Day accuracy of 86.8% (SD=5.6%) and average Work-Time accuracy of 72.1% (SD=9.2%). These results show that Mode 1 does an adequate job at predicting future spatiotemporal events, but the performance decreases greatly during Work-Time. Depending on the application domain this might be good enough, but one has to weigh the increased computational complexity with the performance gain. One advantage that Mode 1 has over Mode 2 and PreHeat is its independence from prior daily spatiotemporal events. Mode 1 makes predictions solely based on the current state and not based on any previous states. That gives the algorithm flexibility in cases for which a spatiotemporal event history is not available (*e.g.*, equipment failure).

ABC-Pattern-Predict Energy Simulation Performance

As we were able to see in the discussion of the algorithmic results, ABC-Pattern-Predict has a very high performance. One aspect of this dissertation is also to highlight how our algorithm impacts application domains. For our analysis we chose efficient room-level temperature control in large office buildings. In order to be able to analyze this domain from different perspectives we chose to use commercially available energy modeling software. Our results were drawn from two different temperature situations (Pittsburgh and Los Angles) and seven different temperature setbacks (2°C to 14°C). As we were able to see in the results

that the energy consumption of the building decreased with increasing setbacks. This was a behavior we were expecting to see. Due to the temperature conditions, we observed that Pittsburgh primarily sees a decrease in heating consumption, while the Los Angles results show a decrease in cooling consumption. It is interesting to note that the LA results show a linear relationship between setback and consumption.

Since we were also interested to see how our approach compares to a reactive and static schedule based system we calculated the consumption savings by temperature setback. As we can see, we need an setback of at least 6°C (a heating setpoint of 61°F) to see a consumption reduction over the two more simpler control strategies. Since our algorithm's performance is very high we expect that an even higher setback is achievable. One interesting aspect is the near identical performance of the reactive and static schedule based system. The results of our algorithm simulation show that this is not an expected situation, but rather an anomaly. The simulated energy consumption of a building (or a set of rooms) depends on several factors: the physical layout of the building, the occupancy traces provided to EnergyPlus, the set temperature during occupied and unoccupied times, the outside conditions, and the HVAC equipment used to maintain a certain temperature (*e.g.*, floor heating, forced air, etc.). In our experiments we varied two conditions: occupancy traces and temperature distribution. Between the static and reactive schedule there are indeed observable differences for the cooling consumption, but by chance the heating consumption is nearly identical. It is possible that the combination of occupancy traces, temperature distribution, outside temperature conditions and HVAC heating equipment created a situation that is so similar between the two conditions that the results are virtually identical. It is hard to further quantify this since EnergyPlus does not provide detailed estimated temperature traces.

The biggest advantage of using ABC-Pattern-Predict for efficient temperature control is its high Recall performance. As mentioned earlier high Recall results in a lower number of False Negatives, which impacts occupant comfort. As we were able to see in Figure 30, ABC-Pattern-Predict only has 19.7 minutes of average

daily MissTime. This is an improvement of 31.7 minutes over PreHeat. This reduction will allow us to deviate greatly from the preferred temperature and will also result in less energy consumption compared to PreHeat. For example let us assume a simple temperature change model that is able to change the room temperature by 2°C over a 30-minute period. Since the average difference between the algorithms is 31.7 minutes we can use a 2°C higher setback, while maintaining the same comfort level as PreHeat. As we are able to see in Figure 29, if we move from 6°C to 8°C we realize a 16.1% in average savings. This means that while the comfort impact of the two algorithms is the same, our algorithm can save 16.1% more in energy. This is a huge environmental and economic impact. As mentioned, we used the metric of MissTime before [136]. However, since EnergyPlus does not provide precise simulated room-level temperature traces as an output we needed to use a more narrow definition of MissTime, by assuming that an exact temperature needs to be achieved. Normally a comfort temperature bound of ±2°C is assumed, which means our analysis slightly overestimates the comfort impact. This means our algorithm has an even lower impact on human comfort. The differences we see between ABC-Pattern-Predict and PreHeat are also to be expected for a more loose definition of MissTime. It is a minor limitation of this analysis, but we believe acceptable given the flexibility EnergyPlus allows us.

This discussion together with Chapter 6 answers the following research questions: **how well does the detected structure in a person's spatiotemporal behavior allow us to make predictions for future events (Q2)** and **how do the characteristics of the prediction results impact our ability to realize the above-mentioned applications (Q3)**.

8. Conclusion & Future Work

This dissertation explored the use of Conditional Frequent-Pattern Trees for the modeling, understanding, and prediction of human spatiotemporal events and structure. We presented ABC-Pattern-Extract, a FP-Tree based extraction algorithm for spatiotemporal structure, and ABC-Pattern-Predict, a prediction algorithm based on FP-Trees. Frequent-Pattern Trees allow us to consider diverse contextual features, which makes the algorithm *flexible* enough to explore not only patterns of individual behavior, but also relationships between contextual features such as people or spaces. Our algorithm showed that we can extract FP-Trees with an average depth of 5.3 in a reasonable amount of time on a standard laptop without any high-level parallelization concepts such as Hadoop. ABC-Pattern-Predict uses these trees to predict future spatiotemporal events and we demonstrated that it does so with a very high performance. Compared to the previous state of the art algorithm, PreHeat, ABC-Pattern-Predict improves the average accuracy by 2.93%, while also improving the recall by 10.11%. The improvements in accuracy and recall have huge implications for our chosen application domain, efficient room-level temperature control. We demonstrated that it is possible to save up to 16.1% in energy over PreHeat while maintaining occupant comfort. The improvements in Recall translated into a reduction in MissTime (the time a space was not at the preferred temperature) by on average 31.7 minutes.

This dissertation made the following contributions:

1. It presents an algorithm, ABC-Pattern-Extract, capable of explicitly modeling the complex human indoor spatiotemporal structure in a time and space efficient manner, so it can run on a standard laptop without high-level parallelization (*e.g.*, Hadoop) taking less than an hour per user. Doing so will allow us to predict future spatiotemporal events and in turn enable the realization of aforementioned applications.

2. Through ABC-Pattern-Predict it shows how the modeled structure can be used to accurately predict future spatiotemporal events. The results show that it is more accurate than current state of the art algorithms and also greatly improves on the recall error metric.
3. To show the impact of the algorithm on an application domain this dissertation uses office temperature control as an example. Compared to previous state of the art algorithms, ABC-Pattern-Predict improves the potential for energy savings by 16.1% over previous state of the art techniques, while maintaining the overall human thermal comfort.

Future Work

Even though we did not explore the use of ABC-Pattern-Predict for other application domains, we believe it is widely applicable. We designed the algorithm to use different types of context, which will allow it to model structured temporal behavior in many different domains. For example, we can use ABC-Pattern-Extract to model a person's comfort and comfort ratings. We believe that our algorithm is flexible enough to adjust to different application domains. One avenue of extending our approach is to test it on other domains to see how the improvements in recall and accuracy impact the results. This will be especially interesting for applications such as *ad-hoc* meetings. Conceptualizing the errors made by our algorithm will allow us to determine if a reduction in false negatives has the same impact on other domains as it had on efficient temperature control.

Even though we modeled occupancy behavior with our algorithm we can also model more precise location information, for example indoor location data provided by an indoor location tracking system. Further improvements of ABC-Pattern-Predict can be made by using a more fine-grained indoor location dataset and exploring the differences in performance when using that dataset instead of an occupancy tracking dataset. We believe that the performance will be similar to the occupancy case. Additionally, modeling the spatiotemporal structure of all occupants of a space allows to create compound schedules for multi-occupant offices, which is another interesting area of future improvements for our algorithm.

The simulation of energy consumption and comfort offers us a test bed to evaluate the performance of our prediction algorithm, but it also limits us since human thermal comfort is highly individual. Running a longitudinal study that explores how the algorithm reacts under real-world conditions, would allow an evaluation how our algorithm reacts under real world conditions. It would be especially interesting to see the interaction between various different temperature setbacks and human thermal comfort. For our experiments we assumed a very tight comfort bound, which overestimates the comfort impact. We believe that in a real world situation it is possible to even further deviate from a person's preferred temperature.

ABC-Pattern-Extract was used to predict spatiotemporal events, but we designed the algorithm on purpose as a standalone. There are two possible applications we can realize with this: 1) explain a person's behavior to themselves and 2) explain the outcome of a prediction to the user. Because the algorithm directly models the spatiotemporal structure it is possible to transcribe the individual patterns in plain language. This will allow users to reason over these pattern and support applications such as intelligibility [45]. Additionally the extraction algorithm allows researchers to explore the structure found in a person's behavior, which makes it possible to not only explain the results of prediction algorithms, but also improve or offer new services.

Besides exploring different application domains, extending the algorithm to more fine-grained spatiotemporal datasets, running field studies with our algorithm, or intelligibility support, we can also improve the algorithm itself.

First and foremost, to be useful in a real-world scenario it is necessary to create a method to update the FP-Trees as soon as new knowledge comes in. Using the same extracted behavior model for several month or even years will inevitably lead to a degradation of the algorithm's performance. A person's schedules is subject to changes over time, which is also why we included a season and day of year contextual feature in the original description. The decision *when* to update the FP-Tree base is complex and requires additional research. For example the deviation from an expected result might be either a unplanned infrequent exception or the

start of a completely new routine that overrides an older established one. Finding a metric (*e.g.*, frequency of occurrence of new routine or fixed temporal length) that makes a decision to update the FP-Tree base is part of the future work.

Another area of future work is to choose the *right* value for the frequency threshold in step 2 of ABC-Pattern-Extract. The frequency threshold has consequences on the computational complexity of the algorithm as well as the expressiveness of the model. By choosing a lower threshold the model becomes more expressive and can potentially react to more situation (esp. for the MLE prediction), but it increases the computational complexity. On the other hand choosing a higher threshold makes the model less expressive, but lowers the computational complexity. Finding the right balance between these two competing goals can further improve the algorithm's performance.

Determining if an event is frequent or infrequent given the user and the current situation is a challenging problem. ABC-Pattern-Predict makes a decision based on the frequency of Singular Patterns using the relative frequency between the two primary predictors. We believe it is possible to further improve the predictive performance by defining a more precise way of determining when an event is frequent or infrequent and subsequently when to use one predictor over the other.

Another way of improving the prediction performance is to consider not just the absolute probability of patterns, but weighing them given other contextual factors (*e.g.,* behavior of another person). This will allow for a more fine-grained decision between several competing patterns.

Lastly to further improve ABC-Pattern-Predict it would be interesting to create a hybrid out of PreHeat and our algorithm. We noted before that PreHeat is capable of reacting to infrequent situations since it is not checking the probability of an event. By also allowing ABC-Pattern-Predict to react to unusual situations we believe that it is possible to further improve the algorithmic performance.

9. References

1. Agarwal, Y., Balaji, B., Dutta, S., Gupta, R.K. and Weng, T. "Duty-cycling buildings aggressively: The next frontier in HVAC control." In Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on, pp. 246-257. IEEE, 2011.
2. Agarwal, Y., Balaji, B., Gupta, R., Lyles, J., Wei, M., & Weng, T. (2010, November). Occupancy-driven energy management for smart building automation. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building* (pp. 1-6). ACM.
3. Anhalt, J., Smailagic, A., Siewiorek, D.P., et al. Toward context-aware computing: experiences and lessons. *IEEE Intelligent Systems 16*, 3 (2001), 38–46.
4. Ashbrook, D. and Starner, T. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing 7*, 5 (2003), 275–286.
5. Aswani, A., Master, N., Taneja, J., Culler, D., & Tomlin, C. (2012). Reducing transient and steady state electricity consumption in HVAC using learning-based model-predictive control. *Proceedings of the IEEE*, *100*(1), 240-253.
6. Autodesk Revit: http://www.autodesk.com/products/revit-family/overview
7. Balaji, B., Teraoka, H., Gupta, R., and Agarwal, Y. "Zonepac: Zonal power estimation and control via hvac metering and occupant feedback." In Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings, pp. 1-8. ACM, 2013.
8. Balaji, B., Xu, J., Nwokafor, A., Gupta, R., and Agarwal, Y. Sentinel: Occupancy Based HVAC Actuation Using Existing WiFi Infrastructure Within Commercial Buildings. *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, ACM (2013), 17:1–17:14.

9. Biswas, J. and Veloso, M. WiFi localization and navigation for autonomous indoor mobile robots. *2010 IEEE International Conference on Robotics and Automation (ICRA)*, (2010), 4379–4384.
10. Brdiczka, O., Su, N. M., & Begole, B. (2009, April). Using temporal patterns (t-patterns) to derive stress factors of routine tasks. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems* (pp. 4081-4086). ACM.
11. Burbey, I. E. (2011). *Predicting future locations and arrival times of individuals* (Doctoral dissertation, Virginia Polytechnic Institute and State University).
12. Clear, A. K., Morley, J., Hazas, M., Friday, A., & Bates, O. (2013, September). Understanding adaptive thermal comfort: new directions for UbiComp. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing* (pp. 113-122). ACM.
13. Cleary, J.G. and Witten, I. Data compression using adaptive coding and partial string matching. *Communications, IEEE Transactions on 32*, 4 (1984), 396–402.
14. Craglia, M., Haining, R., & Wiles, P. (2000). A comparative evaluation of approaches to urban crime pattern analysis. *Urban Studies*, *37*(4), 711-729.
15. Crawley, D. B., Lawrie, L. K., Pedersen, C. O., & Winkelmann, F. C. (2000). Energy plus: energy simulation program. *ASHRAE journal*, *42*(4), 49-56.
16. Davidoff, S. Routine as resource for the design of learning systems. In *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing-Adjunct* (pp. 457-460). ACM.
17. Davidoff, S., Ziebart, B. D., Zimmerman, J., & Dey, A. K. (2011, May). Learning patterns of pick-ups and drop-offs to support busy family coordination. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1175-1184). ACM.
18. Dounis, A. I., & Caraiscos, C. (2009). Advanced control systems engineering for energy and comfort management in a building environment—A review. *Renewable and Sustainable Energy Reviews*, *13*(6), 1246-1261.

19. Erickson, V. L., Achleitner, S., & Cerpa, A. E. (2013, April). POEM: Power-efficient occupancy-based energy management system. In *Information Processing in Sensor Networks (IPSN), 2013 ACM/IEEE International Conference on* (pp. 203-216). IEEE.
20. Erickson, V. L., Carreira-Perpiñán, M. Á., & Cerpa, A. E. (2011, April). OBSERVE: Occupancy-based system for efficient reduction of HVAC energy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on* (pp. 258-269). IEEE.
21. Erwig, M. (2004). Toward Spatio-Temporal Patterns. In *Spatio-Temporal Databases* (pp. 29-53). Springer Berlin Heidelberg.
22. Fischer C. Feedback on household electricity consumption: a tool for saving energy? *Energy Efficiency* 1, 2008
23. Fischer, C. and Gellersen, H. Location and navigation support for emergency responders: A survey. *IEEE Pervasive Computing 9*, 1 (2010), 38–47.
24. Fisk, W. J. (2008). A pilot study of the accuracy of CO2 sensors in commercial buildings. *Lawrence Berkeley National Laboratory*.
25. Gluck, J., Koehler, C., Mankoff, J., Dey, A., Agarwal, Y. A Systematic Approach for Exploring Tradeoffs in Predictive HVAC Control Systems for Buildings. *2nd ACM International Conference on Embedded Systems For Energy-Efficient Built Environments* [In Submission]
26. Gockley, R., Bruce, A., Forlizzi, J., et al. Designing robots for long-term social interaction. *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, IEEE (2005), 1338–1343.
27. Gomes, R., Welling, M., & Perona, P. (2008). Incremental learning of nonparametric Bayesian mixture models. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on (pp. 1-8). IEEE.
28. Gupta, M., Intille, S. S., & Larson, K. (2009). Adding gps-control to traditional thermostats: An exploration of potential energy savings and design challenges. In *Pervasive Computing* (pp. 95-114). Springer Berlin Heidelberg.

29. Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, *8*(1), 53-87.
30. Hilsenbeck, S., Bobkov, D., Schroth, G., Huitl, R., and Steinbach, E. Graph-based data fusion of pedometer and WiFi measurements for mobile indoor positioning. *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM (2014), 147–158.
31. Hnat, T. W., Srinivasan, V., Lu, J., Sookoor, T. I., Dawson, R., Stankovic, J., & Whitehouse, K. (2011, November). The hitchhiker's guide to successful residential sensing deployments. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems* (pp. 232-245). ACM.
32. Hoerling, M. P., & Kumar, A. (2002). Atmospheric response patterns associated with tropical forcing. *Journal of Climate*, *15*(16).
33. Hydeman, M., California Energy Commission, and others. *Advanced Variable Air Volume: System Design Guide: Design Guidelines*. California Energy Commission, 2003. Print.
34. Iso, E. (2005). 7730: 2005:“Ergonomics of the thermal environment–Analytical determination and interpretation of thermal comfort using calculation of the PMV and PPD indices and local thermal comfort criteria”. *International Organization for Standardisation, Geneva*.
35. Karjalainen S. and Vastamäki R. Occupants have a false idea of comfortable summer season temperatures. In: Seppänen O , Säteri J eds. Proc. of *Climate 2007 WellBeing Indoors*. Helsinki: FINVAC - The Finnish Association of HVAC Societies, 2007
36. Kleiminger, W., Mattern, F., & Santini, S. (2014). Predicting household occupancy for smart heating control: A comparative performance analysis of state-of-the-art approaches. *Energy and Buildings*, *85*, 493-505.
37. Koehler, C., Banovic, N., Oakley, I., Mankoff, J., Dey, A., "Indoor-ALPS: An Adaptive Indoor Location Prediction System", In Proceedings of the 2014 ACM international joint conference on Pervasive and ubiquitous computing. ACM

38. Koehler, C., Ziebart, B.D., Mankoff, J., and Dey, A.K. TherML: occupancy prediction for thermostat control. *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, ACM (2013), 103–112.
39. Krumm, J. and Horvitz, E. Predestination: Inferring destinations from partial trajectories. In *UbiComp 2006: Ubiquitous Computing*. Springer, 2006, 243–260.
40. Kuncheva, L. I., Bezdek, J. C., & Duin, R. P., Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, *34*(2), 299-314.
41. L. Knopoff, A. Gabrielov, and M. Ghil, editors. IMA Workshop on Spatio-Temporal Patterns in the Geosciences, University of Minnesota, Minneapolis, MN, USA, September 2001.
42. Landis, J.R. and Koch, G.G. The measurement of observer agreement for categorical data. *biometrics*, (1977), 159–174.
43. Lee, K. C., & Cho, H. (2010). Performance of Ensemble Classifier for Location Prediction Task: Emphasis on Markov Blanket Perspective. *International Journal of U-& E-Service, Science & Technology*, *3*(3).
44. Lee, S., Lee, K.C., and Cho, H. A Dynamic Bayesian Network Approach to Location Prediction in Ubiquitous Computing Environments. In *Advances in Information Technology*. Springer, 2010, 73–82.
45. Lim, B. Y., & Dey, A. K. (2011, September). Investigating intelligibility for uncertain context-aware applications. In *Proceedings of the 13th international conference on Ubiquitous computing* (pp. 415-424). ACM.
46. Lu J., Sookoor T., Srinivasan V., Ge G., Holben B., Stankovic J., Field E., Whitehouse K. The Smart Thermostat: Using Occupancy Sensors to Save Energy in Homes. Proc. Of the *8th ACM Conference on Embedded Networked Sensing Systems* (SenSys), 211-224, 2010
47. Masoso, O. T., & Grobler, L. J. (2010). The dark side of occupants' behaviour on building energy use. *Energy and Buildings*, *42*(2), 173-177.

48. Mathews, E. H., Botha, C. P., Arndt, D. C., & Malan, A. (2001). HVAC control strategies to enhance comfort and minimise energy usage. *Energy and Buildings*, *33*(8), 853-863.
49. McNett, M. and Voelker, G.M. Access and mobility of wireless PDA users. *ACM SIGMOBILE Mobile Computing and Communications Review 9*, 2 (2005), 40–55.
50. Mielke, H. W. (1989). *Patterns of life: biogeography of a changing world*. Boston: Unwin Hyman.
51. Mynatt, E. and Tullio, J. Inferring calendar event attendance. *Proceedings of the 6th international conference on Intelligent user interfaces*, ACM (2001), 121–128.
52. Oh, S. Using an Adaptive Search Tree to Predict User Location. *JIPS 8*, 3 (2012), 437–444.
53. Peffer T., Pritoni M., Meier A., Aragon C., Perry D., How people use thermostats in homes: A review. B*uilding and Environment 46,* 2529-2541, 2011
54. Pérez-Lombard, L., Ortiz, J., & Pout, C. (2008). A review on buildings energy consumption information. *Energy and buildings*, *40*(3), 394-398.
55. Petzold, J., Bagci, F., Trumler, W., and Ungerer, T. Comparison of different methods for next location prediction. In *Euro-Par 2006 Parallel Processing*. Springer, 2006, 909–918.
56. Petzold, J., Bagci, F., Trumler, W., and Ungerer, T. Global and local state context prediction. *Artificial Intelligence in Mobile Systems*, (2003).
57. Petzold, J., Pietzowski, A., Bagci, F., Trumler, W., and Ungerer, T. Prediction of indoor movements using bayesian networks. In *Location-and Context-Awareness*. Springer, 2005, 211–222.
58. Pudil, P., Novovičová, J., & Kittler, J. (1994). Floating search methods in feature selection. *Pattern recognition letters*, *15*(11), 1119-1125.
59. Raudys, Š., & Roli, F. (2003). The behavior knowledge space fusion method: Analysis of generalization error and strategies for performance improvement. In *Multiple Classifier Systems* (pp. 55-64). Springer Berlin Heidelberg.

60. Rosenthal, S., Biswas, J., & Veloso, M. (2010, May). An effective personal mobile robot agent through symbiotic human-robot interaction. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1* (pp. 915-922). International Foundation for Autonomous Agents and Multiagent Systems.
61. Ryan, C. and Brown, K.N. Occupant Location Prediction Using Association Rule Mining. *Workshop on AI Problems and Approaches for Intelligent Environments*, (2012), 27.
62. Scott, J., Bernheim Brush, A.J., Krumm, J., et al. PreHeat: controlling home heating using occupancy prediction. *Proceedings of the 13th international conference on Ubiquitous computing*, ACM (2011), 281–290.
63. Segev, A., Shoshani, A., Logical modeling of temporal data. In *ACM Sigmod Record* (Vol. 16, No. 3, pp. 454-466). ACM.
64. Suzue, T., Shinoda, Y., Highly reproducible spatiotemporal patterns of mammalian embryonic movements at the developmental stage of the earliest spontaneous motility. *European Journal of Neuroscience*, *11*(8), 2697-2710.
65. Teevan, J., Karlson, A., Amini, S., Brush, A.J., and Krumm, J. Understanding the importance of location, time, and people in mobile local search behavior. *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, ACM (2011), 77–80.
66. Trumler, W., Bagci, F., Petzold, J., and Ungerer, T. Smart doorplate. *Personal and Ubiquitous Computing 7*, 3-4 (2003), 221–226.
67. U.S. Census Bureau *Commuting in the United States* (http://www.census.gov/prod/2011pubs/acs-15.pdf), 2009
68. U.S. Department of Energy *Central Air Conditioning Life Cycle Cost Calculator*, http://www.energystar.gov/ia/business/bulk_purchasing/bpsavings_calc/Calc_CAC.xls

69. U.S. Energy Information Administration (eia): Residential Energy Consumption Survey (RECS): End-Use Consumption (http://www.eia.gov/consumption/residential/data/2009), 2013
70. US Department of Energy. Buildings Energy Data Book. http://buildingsdatabook.eren.doe.gov/, Aug. 2012.
71. Vintan, L., Gellert, A., Petzold, J., and Ungerer, T. Person movement prediction using neural networks. *First Workshop on Modeling and Retrieval of Context*, (2004).
72. Voigtmann, C. and David, K. A Survey To Location-Based Context Prediction. *First Workshop on recent advances in behavior prediction and pro-active pervasive computing*, (2012).
73. Vuong, N.K., Chan, S., Lau, C.T., and Lau, K.M. A predictive location-aware algorithm for dementia care. *Consumer Electronics (ISCE), 2011 IEEE 15th International Symposium on*, IEEE (2011), 339–342.
74. Weng, T., & Agarwal, Y. (2012). From buildings to smart buildings—sensing and actuation to improve energy efficiency. *IEEE Design & Test of Computers*, *29*(4), 36-44.
75. Wong, J. K. W., Li, H., & Wang, S. W. (2005). Intelligent building research: a review. *Automation in construction*, *14*(1), 143-159
76. Ziebart, B.D., Maas, A.L., Dey, A.K., and Bagnell, J.A. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. *Proceedings of the 10th international conference on Ubiquitous computing*, ACM (2008), 322–331.

10. Statement of Attribution

The content of this dissertation document is based on published and unpublished material that was created by the author and his collaborators. Chapter 2 and 3 are based on a journal paper currently in submission at Transactions on Intelligent Interactive Systems and is solely written by the author. Chapter 4 was previously published in UbiComp 2014 [37] and is largely based on work by the author. The statistical analysis in 4.3 was performed and written up by the author's collaborator Nikola Banovic. Chapter 5 is the original work of the author (except were explicitly referenced). The simulation framework described in 6.1 was created in collaboration with Joshua Gluck. The author was responsible for the analysis of the results (including the creation of the appropriate Revit and EnergyPlus models) and part of the write-up. The occupancy simulation framework and most of the writing was being done by Joshua Gluck. The analysis of the results of ABC-Pattern-Predict was done by the author.