2018

# A data-driven situation-aware framework for predictive analysis in smart environments

Hoda Gholami
*Iowa State University*

**A data-driven situation-aware framework for predictive analysis in smart environments**

by

**Hoda Gholami**

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
Carl K. Chang, Major Professor
Pavan Aduri
Samik Basu
Jennifer Margrett
Jin Tian

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2018

## DEDICATION

I dedicate my thesis to my family and many friends who have supported me all the way since the beginning of my studies. A special feeling of gratitude to my parents who always believe in me and have made sacrifices so I can achieve my goals. To my siblings who have never left my side and have been a great source of motivation and inspiration. To many friends who have supported me throughout the process. I will always appreciate all they have done.

# TABLE OF CONTENTS

Page

# LIST OF FIGURES

Page

# LIST OF TABLES

Page

# NOMENCLATURE

| | |
|---|---|
| IoT | Internet of Things |
| OpenSHS | Open Smart Home Simulator |
| LSH | Locality Sensitive Hashing |
| CRF | Conditional Random Field |
| SSP | Semi-Supervised PageRank |
| HHSMM | Hierarchical Hidden Semi-Markov Model |
| HMM | Hidden Markov Models |
| NN | Nearest Neighbor |
| KNN | K-nearest Neighbors |
| ANN | Approximate Nearest Neighbor |

## ACKNOWLEDGMENTS

I would like to thank my committee chair, Professor Carl K. Chang, and my committee members, Professor Pavan Aduri, Professor Samik Basu, Professor Jennifer Margrett, and Professor Jin Tian, for their guidance and support throughout the course of this research.

In addition, I would also like to thank my friends, colleagues, the department faculty and staff for making my time at Iowa State University a wonderful experience.

## ABSTRACT

In the era of Internet of Things (IoT), it is vital for smart environments to be able to efficiently provide effective predictions of users' situations and take actions in a proactive manner to achieve the highest performance. However, there are two main challenges. First, the sensor environment is equipped with a heterogeneous set of data sources including hardware and software sensors, and oftentimes complex humans as sensors, too. These sensors generate a huge amount of raw data. In order to extract knowledge and do predictive analysis, it is necessary that the raw sensor data be cleaned, understood, analyzed, and interpreted. The second challenge refers to predictive modeling. Traditional predictive models predict situations that are likely to happen in the near future by keeping and analyzing the history of past users' situations. Traditional predictive analysis approaches have become less effective because of the massive amount of data that both affects data processing efficiency and complicates the data semantics. In this study, we propose a data-driven, situation-aware framework for predictive analysis in smart environments that addresses the above challenges.

# CHAPTER 1. INTRODUCTION

The term 'Internet of Things' (IoT) was first used by the author in [1]. He stated "*The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so*". IoT will comprise billions of sensors, actuators and the data processors. These computational resources enable sensing, capturing, cleaning, organizing and processing of real time data from a vast array of devices serving many different applications including environmental monitoring, industrial applications, business and human-centric pervasive applications [2], [3]. In IoT environment, the user has access to different services at any time and any place. In fact, the devices and network services coordinate with each other to help the user accomplish her tasks without much interference. There are many definitions for IoT. [4] defines IoT as the growing and largely invisible web of interconnected smart objects that promises to transform the way we interact with everyday things. Definition provided by [5] covers broader aspects of IoT, "*The Internet of Things allows people and things to be connected Anytime, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service*". We apply the definition in [5] due to its broader scope.

Due to recent developments in technologies such as sensors, smart devices, cloud/edge/fog computing, and large-scale deployment, the storage and processing power increase and the cost decreases rapidly. As a result, many sources (sensors, humans, applications) start generating big data. Also, organizations tend to store a large amount of data for a long time due to inexpensive storage and processing capabilities. The significant amount of data generated allows IoT applications to make data-driven decisions [6] in a timely manner where money can be saved and operations can be optimized. However, the generated big data may not be of practical values unless it be understood, analyzed,

interpreted in order to extract knowledge. Predictive analysis is expected to do the job. The traditional modeling and reasoning approaches to tackle data have become infeasible because this huge amount of data affects data processing efficiency. Power consumption in such IoT applications is also a key concern. Context-aware computing can play a significant role in addressing such challenges in the IoT paradigm [2], [7].

According to [7], context-awareness is an essential component in loT systems. The term "context-aware" was coined by Schilit and Theimer [8]. The study on context-aware computing has started from desktop applications and has continued to web applications, mobile computing, ubiquitous computing and IoT over the last two decades. There are many definitions for context but in this research, we accept a broader definition provided in [9]. The authors in [9] defined context as "*any information that can be used to characterize the situation of an entity. An entity is a person, place, or an object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*". However, context-aware systems are capable of interpreting raw sensor data, they do not continuously consider human intention that is a core factor in triggering a new behavior[10]. Some studies [11], [10] have proposed human-intention-centric models in context-aware environment. The authors in [11] propose a framework called *Situ* in order to model and predict the user intention which is a vital component of predictive analysis [12]. *Situ* defines a new concept called "situation" as a time-stamped triplet that includes user's desire, behavioral contexts and environmental context values. This definition adds an upper layer to context-aware computing in order to enable situation-aware computing. In order to have an effective mechanism to understand and interpret sensor data, it is important to take into account the factors that have an impact on user's behavior. Human intention,

environmental context values and behavioral context values are tightly interrelated. For example, context values related to previous situation of the user may influence the user's current desire, and the following actions and context values may be determined by the current desire. A change in human's desire usually depends on current context values that may be quite dynamic [11], [10].

In situation-aware computing, the situation data add more meaning and value to sensor data. However, it also increases the storage requirements by many folds. For example, for each sensor log, the meta information such as what properties are measured, which sensors are available, where they are located, what the user's desire is, and what actions taken will be stored.

In this research, we propose a data-driven situation-aware framework for predictive analysis of big situation data in smart environments. We generate situation data using a smart home simulation tool called OpenSHS [13][14]. The collected sensor data has two features that may cause problems in further processing: first, high dimensionality and second, not all of the data is useful for predictive analysis. We take advantage of Scalable Situ Ranking [14] technique to address high dimensionality by filtering out the unimportant situations [12] and utilize only the high value ones in predictive analysis. After preprocessing the data and extracting the knowledge, we build a predictive model using locality sensitive hashing technique. Instead of computing exact similarity between any two situation sessions, we find near neighbors of user's current session in a more efficient way compared to the traditional machine learning techniques. We introduce a new concept called Situ-Morphism that does the task of mapping the *Situ* space to vector space where nearest neighbor search approaches

are defined preserving situation properties. More specifically, the contributions of this study are:

- Proposing a novel data-driven situation-aware predictive framework for predicting user's future situations in a smart environment
- Proposing a scalable ranking technique to rank situations using MapReduce
- Proposing a new concept called Situ-Morphism, a structure-preserving function that maps from situation data to vector data in order to apply LSH technique to numeric vector space using existing methods
- Introducing new similarity metrics in situ-space that measure similarity of situation sessions
- Applying locality sensitive hashing technique to return near neighbors of a given situation session in linear time and efficient space complexity

Although the proposed framework is applicable to any smart environment, we verify our framework on a dataset generated from simulating a smart home using OpenSHS. Smart homes have always been considered as a key part of the ubiquitous computing. Also, we design a small smart home scenario and use it throughout the research to facilitate the understanding of the concepts defined in this study.

# CHAPTER 2. RELATED WORK

## Aware Computing

We have entered the "aware computing" era [14]. We will first make a distinction between context-aware computing and situation-aware computing.

### Context-aware Computing

In the IoT paradigm, there are many sensors that sense and produce low-level context information. It is predicted that the number of smart devices will be over 50 billion by 2020 [15]. The amount of data that will be collected by these sensors is expected to be huge. In order to extract hidden knowledge from big sensor data, IoT systems need to understand and interpret the raw sensor data [16]. Context-awareness helps with interpreting and analyzing data. There are two types of context-awareness: passive and active. In passive context-awareness, system constantly tracks the environment and it helps the user doing her activities by offering the appropriate supports. For example, when a user wants to watch a movie, the mobile phone alerts the user with a list of new movies. However, in active context-awareness, the system continuously and independently observes the user's context and takes an appropriate action. For example, if the smoke detectors and temperature sensors detect a fire in a kitchen in a smart home, the system will automatically make a phone call to inform the resident of the house [7].

### Situation-aware Computing

In the IoT era, more deployment of different types of sensors lead to a rise to different genres of contexts, for example, location, traffic flow, temperature, air quality, etc. What is missing is a mechanism to capture and integrate diverse contextual data collected from various sensors, including humans [17], for enhanced computational intelligence in software

development, maintenance and evolution. This missing mechanism is called “situation”. One of the early studies on situations is the situation calculus which is a form of logic created to represent and reason about dynamic domains. Situation calculus is first introduced by John McCarthy in 1969 [18] but the main version that is commonly used today is introduced by Ray Reiter in 1991 [19]. The concept of situation is given plentiful meanings in several situation models [20]. John McCarthy proposes that a situation is the complete state of the universe at a time instant, while Ray Reiter gives situation a new definition that represents a history of action occurrences. According to Ray Reiter’s work, a dynamic world is modeled as progressing through a series of situations as a result of various actions performed within the world. And situation is defined as a set of contexts in a system over a period of time that has an impact on future behavior of system in specific domains, while context refers to any noticeable property of the environment, the system, or the users in a time instance. In situation theory proposed by Keith Devlin [21], situation refers to parts of the world consisting of objects, their properties and relations to one another, that can be distinguished in common sense. In the IoT era, where humans can be considered as sensors, it is imperative to consider human’s mental state, such as human’s desire or emotion, as another type of context which is usually hidden. As such, we consider situation-aware computing as different from the commonly studied context-aware computing that the human’s mental context becomes an integral part of the underlying computational model. *Situ* [11] defines situation as a triplet of user’s desire, behavioral context and environmental context values at a time instant. This definition is rich in semantics and it helps with modeling and reasoning human mental state. The advantages of this definition are: first, context values give meaning to the user’s actions. Second, user’s intention to reach a certain goal can be inferred based on

observing behavioral and environmental context values. Third, the changes in user's intention can be detected through monitoring sequences of situations for achieving a certain goal. We adopt the definition provided in *Situ* [11] since it combines user's desire with other commonly used context parameters to add an upper layer to context-aware computing. *Situ* is the first computational framework that is capable of modeling and reasoning human intentions by extracting user's hidden desires [10]. The user's behavior is an observable variable that is affected by user's mental state which is an unobservable variable. User's desire can be inferred as reported in [10] where CRF is used.

## Smart Home and its Applications

IoT has the potential to make the development of a wide range of applications possible, however only a very small part is currently available to our society. Many of these applications would likely improve the quality of our life: at home, while traveling, when sick, at work, when jogging and at the gym, etc. These environments are currently equipped with objects with only basic intelligence, and mostly without any communication capabilities. Giving these objects the capability of communicating with other objects to enrich the information perceived from the surroundings would enhance the smartness of environments where a very large array of IoT applications can be deployed [22].

The advances of sensor technology together with the increase in power of computation has resulted in rapid emergence of smart environments such as smart homes [23]. Smart homes first described by Weiser [24] on a small scale, are "*improved living spaces equipped with distributed sensors and effectors hidden from the view of the residents*". Smart homes are designed to enhance the quality of life of residents by automating daily tasks and optimizing power consumption. In addition to enhancing quality

of life, smart home technology provides a mechanism to assist the residents in their daily life activities [25]. To achieve that goal however, many challenges such as activity recognition, emotion recognition, abnormal behavior detection and predicting the diseases before they happen need to be addressed [26]. Activity recognition is an old and well-studied problem. There are mainly two approaches for activity recognition: knowledge driven and data driven. Probabilistic and logic based activity recognition algorithms are considered as knowledge driven approaches. The main limitation of knowledge driven approaches is that it is assumed that the knowledge of the observing agent regarding the human participant mainly to be complete and correct. This knowledge is usually encoded in a complex formalism such as first order logic that requires a significant computer science expertise. Data-driven approaches have emerged to overcome the limitations related to the knowledge library required by the knowledge driven approaches. Data-driven approaches are suited very well for this study to tackle the emerging big data aspect of smart homes [25].

Context-awareness is often considered as complementary component to activity recognition since it helps with sensing and understanding of the activity and location. Therefore, it is desirable for smart homes to be characterized as a context-aware application [25]. Different types of sensors such as motion sensors, door entry-point sensors, taps, cooker sensors, pressure sensors, kettles, etc. capture different context values that would help to identify patterns of user's behavior and eventually detect and categorize user's behavior. After learning the user's behavioral patterns, it becomes feasible to detect any abnormal behavior.

There exist many other studies on smart homes with a variety of applications such as monitoring systems for elderly independent living, accident and fall detection, etc. [27], [28],

[29]. Many other studies focus on addressing the issues of independent living in a broader sense [30], [31], [32], [33]. There are also researchers who study identifying and modeling progression of specific diseases such as dementia of the Alzheimer's type by monitoring performance in the execution of daily tasks [26], [34].

Several methods including neural networks [26], [35], [36], [37], [38], [39], knowledge driven [40], [41], fuzzy logic [27], [28], [29], time series analysis [42], heuristic and machine learning techniques [43], [44], [45], [40], [37] are utilized to predict user's behavior by observing daily activities and capturing the behavioral patterns. Some studies [12], [46], [47] take advantage of probabilistic and statistical analysis techniques since sensor data are noisy and human activities are performed in a complex nondeterministic mode. The issue with these methods is that they usually require large training data set in order to create an acceptable reasoning model. Also, there is no guarantee for detecting an abnormality condition. However, some studies [12] have partially addressed some of the issues related to these techniques such as high computational complexity of probabilistic and statistical models.

Therefore, in the context of smart homes, there is a demand to have a scalable intelligent mechanism to monitor user's daily activities, infer the user's mental state, extract the behavioral patterns and create the predictive model to provide predictions and recommendations with respect to user's current desire.

## Scalable Situ Ranking: Filtering Situations

### PageRank

PageRank [48] is the most popular link analysis algorithm, and it is mostly used for assigning numerical weights (i.e., ranks) to web documents in a web graph. Rank of a node is

correlated with the importance of that node, as it is defined based on the number and the importance of the nodes linking to it. More specifically, the PageRank of a web page is defined as the probability of the random surfer being at some particular time step k > K at this web page. Nodes with high in-degrees are more likely to have higher ranks, so are nodes that are linked to by other high ranked nodes. This probability is more likely to remain same for a large enough K. There are many studies using PageRank in its original form (i.e. jumping to a random node with equal probabilities) or personalized form (i.e. favoring certain nodes) [49], [50], [51], [52], [53]. Formally, the PageRank of a page n is defined as follows:

$$PR(n) = (1-\alpha)\left(\frac{1}{|G|}\right) + \alpha \sum_{m \in In(n)} \frac{P(m)}{|Out(m)|} \qquad (1)$$

Where |G| is the total number of nodes in the graph, α is the damping factor which usually takes 0.85 as its value. In(n) is the set of nodes linking to node n, and |Out(m)| is the out degree of node m.

PageRank infers the importance of a node only based on graph structure. In smart environments, situation graphs can be large with millions of situation nodes with a large amount of information that is accumulated on the graphs. The information includes user behavior data and rich metadata associated with the graphs. Similar to the premises assumed by PageRank, in situation-aware smart environments, the history of user's behavior has an important impact in user's future behavior.

**MapReduce and its Applications**

In the era of big data, parallel processing is vital for processing of data in a timely manner. MapReduce is one of the most recent computing models for large data processing over distributed systems [54]. The main idea of the MapReduce model is to hide details of

parallel execution and let users design data processing strategies [55]. MapReduce was originally studied by Google [56] and has its roots in functional programming with two higher order functions map and reduce. In the first stage a user-specified computation (i.e. map task) is applied over a large number of records in a parallel mode to generate the intermediate results. Then, the results are aggregated by another user-specified computation (reduce task). Mapper and Reducer have the following signatures:

map: (k1, v1) → [(k2, v2)]

reduce: (k2, [v2][1]) → [(k3, v3)]

The basic data structures are in the form of (key, value) pairs. The "mapper" is applied to every input key/value pair to produce an arbitrary number of intermediate key/value pairs. The "reducer" is applied to all intermediate pairs associated with the same key to generate output key/value pairs [57]. Figure 2.1 illustrates the MapReduce framework.

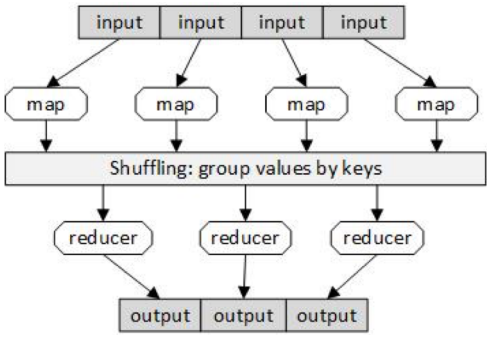



Figure 2.1 MapReduce framework [58]

MapReduce has been applied in many areas such as machine learning [58], [59], [60], text processing [61], [62], [63], [64], bioinformatics [65], [66] and graph-based computations [67],[57],[68] in order to tackle large-data problems with high efficiency. In [68], the authors proposed a scalable Label Propagation algorithm for semi-supervised classification and graph

[1] The convention [..] refers to a list

ranking using MapReduce. They showed how Label Propagation can be utilized for node ranking in graphs. They utilized MapReduce to convert the representation of the graph from an edge format to their desired format. Also, they applied MapReduce for implementing Label Propagation in an iterative manner. The authors in [67] proposed a semi-supervised learning framework for ranking on large-scale graphs called Semi-Supervised PageRank (SSP) using MapReduce. SSP considers meta-information on graphs and supervision information from humans besides the graph structure in graph ranking. They divided the large-scale computation in SSP into two parts: matrix-vector multiplication and Kronecker product of vectors. Then they applied MapReduce to handle the corresponding computations. In [57], the authors focus on general graph algorithms in MapReduce. Graph algorithms that can be applied to solve real world problems include graph search and path planning, graph clustering, minimum spanning trees, bipartite graph matching, maximum flow and identifying "special" nodes. A common property of all these problems is the large-scale datasets, where algorithms that run on a single machine are not scalable. They apply MapReduce to address these challenges. More specifically, the usage of MapReduce algorithms for single-source shortest path, breadth-first search and PageRank are discussed in [57]. There are plenty of other applications that utilize MapReduce to parallelize the computing such as web search, building inverted indexes and IoT [69]. In IoT, MapReduce is useful for processing sensor data in real time. In this study, we utilize MapReduce in order to implement the proposed Scalable Situ ranking technique in a parallel mode.

## Predictive Analysis

The general idea of predictive analysis is to make predications based on observations by matching the current observation with a data set and find the closest pattern to the current one and provide predictions.

### Probabilistic Graphical Models

Over the past decade, many researchers have explored probabilistic graphical models to detect and predict user activities. Markov models [70] have been used for studying and understanding stochastic processes and shown to be well suited for modeling and predicting a user's next action on a graph-based model of user's sequential behavior. In these problems, the input usually is the sequence of observations and the goal is to build Markov models in order to predict the next state of the system. For example, in the problem of web surfing, the input is sequence of web pages accessed by a user, and the output is the web page which is most likely to be accessed by the user next. The authors in [44] use Hierarchical Hidden Semi-Markov Models (HHSMMs) to predict the inhabitant activity. In [43], Hidden Markov Models (HMMs) have been employed to model the residents' behavior in order to detect abnormal activities. Furthermore, in [71], the authors present an activity prediction model based on Markov models and partial matching in the context of smart homes. The authors in [72][72][73] propose an approach based on Markov Logic Networks for decision making in smart homes. They assume that the user interacts with the system through voice and explicitly express her demand. In the context of smart environments, the next situation of a user relies not only on the current situation but also on the history of her milestone activities in the past. Therefore, a predictive probabilistic graphical model which keeps the history of user's activities and support randomness and uncertainty is needed. Higher-order Markov models have been widely used in predicting user's behavior since lower-order models do not

look far into the past in order to predict user's next step. By taking into account the sequence of user behavior, the model would be able to correctly recognize differences between observed patterns and make the most accurate prediction [74][75]. In these studies, the initial probabilities of nodes have a uniform distribution leading to a less accurate prediction results. To address this issue, some other studies [76][77][[74][78][79] [80] have defined different priors in Markov models. For instance, in [76] the authors used a revised version of PageRank algorithm and in [77][80] the authors utilized link and citation analysis to assign prior probabilities to nodes based on their importance in the graph to obtain more objective predictions. Furthermore, some studies have proposed methods to improve the accuracy or reduce state complexity of Markov models in the context of predicting the system's next state [77][81][79]. However, the authors in these studies presented methods to address the problem of m-path prediction (i.e. prediction of user behavior in m steps), and such methods have high computational cost [78][82] and the number of parameters need to estimate grows exponentially with the order.

**The Nearest Neighbor Search Problem**

Many of existing detection problems such as activity detection, fall detection, abnormal activity detection, emotion detection, are based on approximate pattern matching. The common problem of finding an object from a search database that is nearest to a query object is called with different names such as similarity search, nearest neighbor search, proximity search, or close object search. The object that is closest to a query object with respect to a similarity metric is called nearest neighbor [83].

**Exact nearest neighbor search**

Given a query q, nearest neighbor of q shown as NN(q), is chosen from a set of items $I = \{i_1, i_2, \ldots, i_N\}$ so that $NN(q) = \text{argmin}_{i \in I}\, \text{dist}(q, i)$, where dist(q, i) is the "distance"

between q and i. A generalization of nearest neighbor problem is a K-NN search, where the goal is to find K-nearest neighbors of given query q (KNN(q)). K-nearest neighbor search indicates the top k nearest neighbors to the query. This is a common technique in predictive analytics to estimate or classify an item based on the majority votes of its neighbors. In R-near neighbor (fixed-radius near neighbor) problem, a revised version of nearest neighbor search, the goal is to find the set of items i that are within the distance C of q, R = {x| dist(q, i) ≤C, i∈I} [84]. These concepts depend on the notion of the distance function. Researchers have formulated several notions of distance, such as Euclidean distance, cosine distance, hamming distance etc., for various application scenarios. In this work, we use (adaptations of) some of the well-studied distance measures.

**Approximate nearest neighbor search**

Finding exact nearest neighbor and R-near neighbor become infeasible in the large-scale high-dimensional cases because computing exact similarity between two objects is computationally expensive when the size of the data set is massive. Therefore, many recent studies proposed approximate nearest neighbor search, and it is shown that this approach works well for many practical similarity search problems in large datasets. The $(1 + \epsilon)$-approximate nearest neighbor search problem, $\epsilon > 0$, is defined as: Given a query q, the goal is to find an item i that $\text{dist}(q, i) \leq (1 + \epsilon)\text{dist}(q, i^*)$ , where $i^*$ is the true nearest neighbor [85].

The randomized nearest neighbor search problem reports the approximate nearest (or near) neighbors with a probability. There are two widely known randomized search problems: randomized c-approximate R-near neighbor search and randomized R-near neighbor search. In randomized c-approximate R-near neighbor, given a query q, the goal is

to find some cR-near neighbor of the query q with probability $1 - \delta$, where $0 < \delta < 1$. In randomized R-near neighbor search the goal is given a query q, to find some R-near neighbor of the query q with probability $1 - \delta$ [84].

A successful approach for approximate similarity search is via Hashing. The basic idea of hashing is to map the data items into a low dimensional subspace (i.e. short code of a sequence of bits). The hashing approach maps the query items to the target items of low dimensionality therefore approximate nearest neighbor search can be efficiently and accurately performed using found target items, since their dimensionality is low the target items are called hash codes. Formally, the hash function is defined as y = h(i) in which y is the hash code and h is the hash function. For a data item i, its low dimensional representation is calculated by applying multiple hash functions on i. For example, if $h_1, h_2, \dots, h_M$ are the hash functions, then i is mapped to an M-dimensional vector $V_i = [h_1(i)\ h_2(i)\ \cdots h_M(i)]^T$ [85]. Now, to find the distance between two original object i and j, we instead look at $V_i$ and $V_j$. To make the nearest neighbor search fast, the vectors $V_i$ and $V_j$ are stored in a hash table such that if $V_i$ and $V_j$ are close to each other, then they are placed into the same bucket.

Here, the purpose of hashing approach using a hash table is to maximize the number of collisions of near neighbor items. Given query q, the items in the bucket h(q) are retrieved as near neighbors of q. These ideas are known as locality sensitive hashing.

***Locality sensitive hashing***

The term "locality sensitive hashing" (LSH) was first introduced in 1998 [85] as a randomized hashing framework for efficient approximate nearest neighbor (ANN) search in large scale high dimensional space. Locality sensitive hashing is based on the definition of LSH family H, a set of hash functions that map similar items to the same hash code with

higher probability than dissimilar items. In particular, having H as a family of hash functions mapping $R^d$ (i.e. a d-dimensional space) to some set S and h is a function chosen from H uniformly. A family H is called $(S_0, cS_0, p_1, p_2)$-sensitive if for any two points x, y $\in R^d$ these two conditions hold:

- if $Sim(x, y) \geq S_0$ then $Pr_H(h(x) = h(y)) \geq p_1$
- if $Sim(x, y) \leq cS_0$ then $Pr_H(h(x) = h(y)) \leq p_2$

Then we say x is $S_0$-near neighbor of point y. In order to find approximate nearest neighbor, $p_1 > p_2$ and $c < 1$ should hold. Other parameters $S_0$ and $\delta$ are none zero positive values. The definition of LSH family is tightly correlated with the type of similarity function of interest [86]. Given two vectors $V_a = \{a_1, \ldots, a_M\}$ and $V_b = \{b_1, \ldots, b_M\}$, some of the popular similarity/distance measures are as follows:

**Euclidean Distance.** The Euclidean distance between vectors $V_a$ and $V_b$ is defined as:

$$dist(V_a, V_b) = \sqrt{(a_1 - b_1)^2 + \cdots + (a_m - b_m)^2} \qquad (2)$$

**Cosine Similarity.** The cosine similarity between $V_a$ and $V_b$ measures the angle between two vectors which equals to $\frac{V_a.V_b}{L_2(V_a) \times L_2(V_b)}$ (3). Here $L_2(V_a)$ is length of vector $V_a$ and equals to $\sqrt{{a_1}^2 + {a_2}^2 + \cdots {a_m}^2}$. Also, $V_a.V_b$ is dot product of $V_a$ and $V_b$ and its value obtained by $a_1b_1 + a_2b_2 + \cdots + a_mb_m$. Note that if two vectors are identical, $V_a$ and $V_b$ are same thus the angle between $V_a$ and $V_b$ is zero, thus the cosine similarity is 1.

**Jaccard Similarity.** The Jaccard similarity between two vectors $V_a$ and $V_b$ is $\frac{V_a.V_b}{[L_2(V_a)]^2 + [L_2(V_b)]^2 - V_a.V_b}$ (4).

**Jaccard Similarity as Set Similarity.** Given two sets Sa and Sb Jaccard similarity of Sa and Sb is defined as $\frac{|S_a \cap S_b|}{|S_a \cup S_b|}$ (5).

Researches on locality sensitive hashing mainly focus on three aspects. The first aspect is developing various LSH families for various distance or similarity measures. For example, p-stable distribution LSH is designed for ℓp distance measure [87], simhash (sign-random-projection) for angle-based distance [88], minhash for Jaccard similarity [89], [90] and many other variants developed based on these basic LSH families [91]. The second aspect is to look into search efficiency of LSH framework (both time and space) to find the most optimized possible LSH family for certain distances and similarities [87], [92], [93], the tight characteristics for a similarity measure to admit an LSH family [88], [94] and so on. The third one focuses on improving the search scheme of the LSH methods, to improve search efficiency [91], [95].

In this study, we utilize LSH in order to find approximate nearest situation sessions to the user's current situation session. However, defining new metrics to measure similarity between situation sessions and a LSH framework that be applicable in *Situ* space remains as a challenge. We introduce a novel concept called Situ-Morphism later in this research to do this job but before that, we define some concepts used in the rest of this study.

# CHAPTER 3. ESSENTIAL CONCEPTS AND ASSUMPTIONS

In this section, first we introduce some basic concepts that is used in this study. Then we review six assumptions proposed for stipulating the scope of application of the framework that is a situation-aware data-driven approach for predictive analysis in smart environments.

## Essential Concepts

We give an overview of the basic concepts used in this study as follows:

***Situ* space**: A *Situ* space is a collection of objects called situations [14]. All the other concepts are defined in *Situ* space.

**Situation**: A situation is an instant status of the environment including environmental context values, and user, including user's behaviors and user's desire. In particular, a situation can be described as a triplet {d, A, E} of user's desire, actions and all environmental context values at a time instant [11].

**Situation Session**: A situation session is a sequence of the user's situations in a chronological order during a specific period of time while goal is not achieved [12].

**Initial Situation**: The first situation of each situation session is called initial situation. When user starts a new desire, the first situation of user is considered as initial situation of that session.

**Temporally ordered**: The situations in a situation session are temporally ordered. For any two consecutive situations in a situation session, the time-point of the latter situation is after that of the former situation.

**Situation transition**: The transition from a situation to another [12][14].

***Situ graph***: A directed, weighted and strongly connected graph constructed from situations and their transitions. Each node represents a situation and each link represents a transition from one situation to one other [12].

**Desire**: The condition or status of the environment in the domain that the user would like to achieve. The desires drive users to perform actions to achieve certain goals [11].

**Goal**: Set of logical conditions (specific context values) that indicates user's desire has been satisfied [14].

**Milestone situations**: The situations that are more likely to be observed to reach the goal g w.r.t. desire d [14]. Milestone situations are obtained by Scalable Situ Ranking [14] and have higher ranks compared to other situations.

**Milestone transitions**: The transitions between milestone situations that are more likely to be observed to reach the goal g w.r.t. desire d. Milestone transitions are the ones with higher frequencies compared to other transitions w.r.t. desire d [14].

## Assumptions

We make assumptions to delimit the scope of the domains in which the proposed framework can be applied.

**Assumption 1:** The domain of the proposed framework is a rational single-user domain. There is only one user active at a time in the smart environment.

However, in real-life applications, there may be more than one user. In this study, we make assumption 1 to keep the scope of this study manageable. Assumption 1 indicates that the desire and actions in one situation belong to the same user. For more than one user, this model can be extended by assigning a user id to situations to separate the situations based on the user, assuming that technologies permit users to be individually identified.

**Assumption 2:** Any changes in the user's desire trigger to a new session. By assumption 2, our framework is able to recognize the start of a new session.

**Assumption 3:** In any situation if the goal is achieved (all the logical conditions w.r.t desire d are satisfied), we mark that situation as end of the session.

By assumption 3, our framework is able to recognize the end of a session.

**Assumption 4:** For each situation session, we assume that the goal is satisfied by end of the session.

According assumption 4, the user's goal is accomplished at the end of each session. This assumption makes our framework to be built on valid behavioral patterns for predictive analysis. Therefore, abnormal non-complete sessions will be detectable in run time.

**Assumption 5:** This approach supports multi-tasking and interleaving tasks while user is involved in only one active situation at a time instant.

Any changes in a desire triggers to the start of a new session. And any situation that indicates a goal is achieved marks the end of the latest open session. Any two situations may interleave as follows:

- Session j happens during session i: for example, while cooking (desire = cooking), user decides to watch TV too (desire = watching TV). After finishing watching TV (goal is achieved), user goes back to kitchen to finish cooking (goal is achieved). Here we extract the included session j as a separate session for desire watching TV. But for desire cooking, we consider watching TV as part of outer session to capture the pattern of multi-tasking w.r.t the starter desire (here cooking).
- Session j starts in session i but continues even after session i ends or vice versa. For example, user starts cooking, while cooking, she decides to chop the potatoes sitting in

couch and watching TV. User goes back to the kitchen to finish up cooking quick and go back to continue watching TV. Here again, each session including the interleaved situations is processed separately.

**Assumption 6:** All desires are atomic; that means if there is no relationship between desires, they are identical or different. There is no composition nor overlapping relationship among desires.

Working with situations as unique data elements keep the cost of computations reasonable. A case with composite situations can be converted to atomic desires with breaking down each desire to smaller ones and then utilize them for this study.

# CHAPTER 4. SITU MORPHISM

## Morphism

Morphism is a structure-preserving function that maps from one structure to another. Preserving the structure happens in some sense, and this sense has to be specified as part of the theory of those particular structures. Examples of structures are measures, algebraic structures, topologies and metric structures. For example, in graph theory, graph morphism refers to a mapping between two graphs that preserves their structure. In other word, it is a function between the vector sets of two graphs that maps adjacent vertices to adjacent vertices. Function f is morphism for graph H and G if for all vertices x and y in G, condition if $(\mathrm{x},\mathrm{y}) \in \mathrm{E(G)} \Longrightarrow \big(\mathrm{f(x)},\mathrm{f(y)}\big) \in \mathrm{E(H)}$ holds. Or in vector space morphism is defined as a function f: $\mathrm{V} \rightarrow \mathrm{W}$ that preserves the vector space structure that are additive structure and multiplication from vector space V to another space W. Particularly, f is morphism if for any two vectors $\mathrm{u},\mathrm{v} \in \mathrm{V}$ $\mathrm{f(u+v)} = \mathrm{f(u)} + \mathrm{f(v)}$ and $\mathrm{f(av)} = \mathrm{af(v)}$ [96].

**Situ-Morphism**

Let Ses = $\{Ses_1, Ses_2, \ldots, Ses_N\}$ be a collection of user's situation sessions observed in the past. Given current situation session of user, the goal in predictive analysis is to answer questions such as how "similar" two situation sessions $Ses_a$ and $Ses_b$ are? given a session $Ses_a$, what are all the situation sessions "similar" to $Ses_a$? find groups of sessions so that sessions within a group are "similar" to each other. In this research, we map the situation sessions in *Situ* space to vectors in vector space while preserving the similarities between any two situation sessions in the original space. Formally, function g is Situ-morphism from *Situ* space S to vector space V ($g: S \rightarrow V$) if for any two $Ses_a$ and $Ses_b$ in *Situ* space, if $sim(Ses_a, Ses_b) > \delta \; then \; sim\left(g(Ses_a), g(Ses_b)\right) > \delta$ where $0 \leq \delta \leq 1$. Here $Ses_a$ in

original space is a situation session and g($Ses_a$) is its corresponding vector with milestone situations as dimensions. We choose similarity as the main property that needs to be preserved in Situ-Morphism because finding closest neighbors involves computing similarity among objects.

**Situation session features**

Before defining the notion of "similarity" in *Situ* space, we explain how we define and extract features of a situation session. Then, we use the extracted features of sessions to define similarity. We guarantee that Situ-morphism preserves the features of situation sessions. An obvious choice for feature is "situation". We assume S = $\{s_1, s_2,..., s_m\}$ is the set of all situations that appear in the session collection Ses and $r_{ij}$ is the rank of situation i in session $Ses_j$. For a given session $Ses_i \in Ses$, a binary representation is obtained by defining $R_{ses_i} = <r_1, r_2,\ldots,r_m>$ $r_{ij}$ as 1 if $s_i$ appears in $ses_j$ otherwise $r_{ij}$ equals 0. It is reasonable to assume that if two sessions are highly similar to each other, then there exists a large set of situations that appear in both sessions. However, some situations appear in many sessions and they do not contribute much to the similarity of sessions. Such situations are called non-milestone situations. Thus, the non-milestone situations are not features of any session. As a result, we remove all non-milestone situations from every session in our session collection. We define the rank vector as $R_{ses_i} = <r_1, r_2,\ldots,r_m>$ and it is obtained using Scalable Situ Ranking discussed in related work. Each element in $R_{ses_i}$ vector, represents the importance of a situation in the given session. Non-milestone situations are the situations that their rank is less than a specific threshold, most likely because they do not contribute to achieving user's goal. Then, two situation sessions $Ses_a$ and $Ses_b$ can be considered similar if the two sessions have a large fraction of milestone situations in common. Even though sometimes two sessions contain exact same milestone situations, they are not

considered similar sessions. The reason is that besides having common milestone situations, the order of milestone situations in a session is important. Therefore, having $MS = \{ms_1, ms_2, ..., ms_k\}$ as set of all milestone situations, we improve the vector representation of situation sessions to milestone-position vector $Pos_{ses_i} = <pos_1, pos_2, \ldots, pos_k>$ where $pos_{ij}$ indicates the relative position of milestone situation i in session j. Here k is the number of milestone situations in sessions collection Ses and k << m.

**Situation session similarity metrics**

We define "similarity" of situation sessions in form of pyramid (

Figure 4.1) instead of one layer of similarity. With the pyramid structure, the sessions with lower degree of similarity, will go to wider and lower layers and then those with higher degree of similarity will be narrowed down to restricted upper layers. In other words, each layer presents a similarity function that takes any two sessions from lower layer as input and compute their similarity. If two sessions are not similar with respect to a pre-defined threshold they are not moved to current layer, otherwise they are. There are five similarity functions. The first layer includes all sessions and does not represent any function.

The layered similarity metrics are defined as follow:

**Similarity Type I** (Common desire): If two situation sessions $Ses_i$ and $Ses_j$ in layer zero (all situation sessions), have same desire, they are considered as similar sessions type I and go to layer 1. In other words, it is more likely that user has same behavioral pattern in sessions with same desire. For two given sessions, if desires are different, they don't go to upper layers, otherwise they go to upper layer to be checked for higher degree of similarity.

**Similarity Type II** (Common milestone situations): Situation sessions with similarity type I at layer 1, are similar type II at layer 2 if they share many milestone situations. In other words, it is

more likely that user follows same behavioral pattern in sessions with same desire and shared milestone situations.

**Similarity Type III** (Common milestone transitions): If two situation sessions $Ses_i$ and $Ses_j$ in layer 2, share milestone transitions, they are considered as similar type III and go to layer 3. In other words, it is more likely that user follows same behavioral pattern in sessions with same desire and shared milestone situations that have same composition order.

**Similarity Type IV** (Common milestone-position): If two similar situation sessions $Ses_i$ and $Ses_j$ type III at layer 3, share milestone situations with approximately same distance from initial situation, they are considered as similar sessions type IV at layer 4. In other words, it is more likely that user follows same behavioral pattern in sessions with same desire, shared milestone situations with same order and distance from initial situation.

**Similarity Type V** (Identical): If two similar situation sessions $Ses_i$ and $Ses_j$ type IV at layer 4, become identical after removing non-milestone situations, they are considered as similar sessions type V at top level 5. In other words, it is more likely that user follows same behavioral pattern in sessions with exactly same composition of milestone situations in terms of number, position and order.

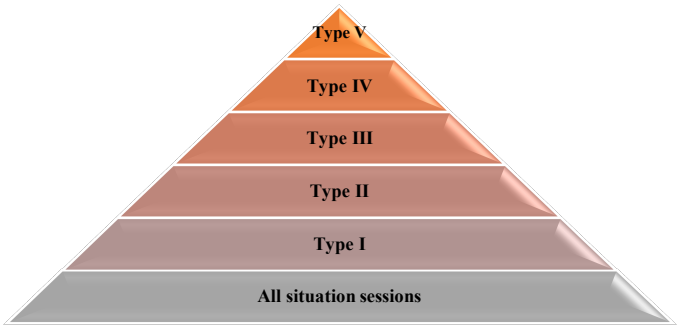



Figure 4.1 Similarity metrics in Situ Space

**Mapping Situ space to vector space**

The morphism function g maps sessions to milestone-position vectors while preserving the milestone situations and their relative position in Situ space. In this part, we explain how Situ-Morphism preserves the similarity between any two situation sessions during mapping process from Situ space to vector space.

***Similarity Type II to Jaccard similarity***

Jaccard similarity coefficient is a similarity measure that compares the similarity between two feature sets. It is defined as intersection of two sets over the size of union of two sets. In situ-space, two sessions are similar Type II if they have some common milestone situations. In vector space, this can be measured with Jaccard similarity. For two sessions with milestone position vectors $A^*$ and $C^*$ , union is the number of milestone situations with non-zero entry in $A^*$ or $C^*$, let's call this value p. And the intersection is the number of milestone situations with non-zero entry in both $A^*$ and $C^*$, let's call this q. In vector space, Jaccard similarity of $A^*$ and $C^*$ is equal to $\mathrm{Sim}_j(\vec{A}, \vec{B}) = \frac{q}{p}$. As the value of this ratio be closer to 1, it means two sessions have more milestone situations in common so they are more similar type II in Situ space and similar in vector space w.r.t Jaccard similarity.

***Similarity Type III to situation-order similarity***

Milestone situations composition plays an important role in understanding of situation sessions and user behavior. Basic composing information, such as situation order, can provide useful information to distinguish two sessions. Without syntactic information, it is impossible to discriminate sessions that share the similar milestones representations. For example, "turn off the water → drying body with towel" and "drying body with towel → turn off the water" will both be classified as similar sessions Type II because they share the

same milestone situations. However, their meanings appear to be different. In the second scenario, user probably has forgotten to turn off the water after shower. Li et al. [84] defines word order similarity measure as the normalized difference of word order between the two sentences. We revise the formula to fit it to domain of this study. The formulation for milestone situation order similarity is as follows:

$$\mathrm{sim_o}(\mathrm{A}^{\rightarrow}, \mathrm{C}^{\rightarrow}) = 1 - \frac{\left|\left|\mathrm{r_{A'}}^{\rightarrow} - \mathrm{r_{C'}}^{\rightarrow}\right|\right|}{\left|\left|\mathrm{r_{C'}}^{\rightarrow} + \mathrm{r_{C'}}^{\rightarrow}\right|\right|} \quad (6)$$

Where $\mathrm{A}'$and $\mathrm{C}'$ are ordered sequences of shared milestone situations in $\mathrm{A}^*$ and $\mathrm{C}^*$. Milestone order vector is a feature vector whose feature set comes from shared milestone situations that appear in a session pair. We create a milestone situation set called T which is a set of all shared milestone situations in A and C. For each shared milestone situation in T, we try to find the same milestone situation in $\mathrm{A}'$and fill the entry for this milestone situation in $\mathrm{r_{A'}}^{\rightarrow}$ with the corresponding index number in $\mathrm{A}'$. We repeat the same process for C. The index number is simply the order number that the milestone situation appears in the set $\mathrm{A}'$ and $\mathrm{C}'$. Therefore, using this formula we can map similarity Type III to order metric in vector space to measure the similarity of two session vectors in layer 2.

***Similarity Type IV to Euclidean distance***

In Situ space, we say two sessions are similar Type IV if their shared milestone situations have approximately same distance from initial situations. In vector space model, we can measure this using Euclidean distance metric. We define a similarity metric as follows:

$$\mathrm{Sim_e}(\mathrm{A}^{\rightarrow}, \mathrm{B}^{\rightarrow}) = 1 - \sqrt{(\mathrm{A_1}^{\rightarrow} - \mathrm{B_1}^{\rightarrow})^2 + (\mathrm{A_2}^{\rightarrow} - \mathrm{B_2}^{\rightarrow})^2 + \cdots + (\mathrm{A_k}^{\rightarrow} - \mathrm{B_k}^{\rightarrow})^2} \quad (7).$$

Table 4.1 Situ-Morphism preserves the similarity between situation sessions

| **Situ Space** | **Vector Space** |
|---|---|
| Similarity Type I | Simple comparison of desires |
| Similarity Type II | Jaccard Similarity |
| Similarity Type III | Situation-Order Similarity |
| Similarity Type IV | 1-Euclidean distance |
| Similarity Type V | Simple comparison of two sessions |

# CHAPTER 5. RESEARCH METHODOLOGY

The problem of predictive analysis in smart environments has two aspects: human aspect and technology aspect.

## Human Complexity

Humans are complex beings, mentally, psychologically, physically and physiologically. It is likely that an end-user may not be able to give well-articulated requirements nor can a software engineer accurately elicit user's requirements. Generally speaking, humans evolve as "situations" arise. In other words, human sensory adaptability to contextual signals from the environment is mapped into mind adaptability to perceived situations. To understand a human and her behavioral pattern, it is necessary to consider mental, emotional, motivational and intentional dimensions of human mental states. Sometimes the situation may become more complex and unpredictable as humans tend to behave differently under different circumstances. For example, in a smart home, to take a shower, there is generally a regular sequence of tasks to complete this goal. However, in one evening the user may prefer to turn on the radio during her shower or brush her teeth, etc. This unusual behavior comes from temporary priorities in user's mind that change the order or type of regular tasks. In order to model user's behavioral context with all such complexities, it is vital to relate the behavioral context to user's mental state by adding user's desire to the model. More precisely, it is important to monitor the user's desire at each time instance and model user's behavior w.r.t the current desire and environmental context values (i.e. situation). This approach helps with more effective predictions because it considers human's complexity, although continuously monitoring human's mental state will largely increase the volume of data collected from the computational framework. Because of the

sizable volume of collecting mental-contextual data, in addition to voluminous environmental and behavioral-contextual data, technological challenges pertaining to big data analytics thus emerge.

## Technology Challenges

Due to many sensors observing user's behavior and producing time-stamp events, predictive analysis approaches in IoT need to provide solutions to address big data challenge. In any smart environment, there are many sensors generating a lot of data. The problem is that this big amount of raw data is not useful until it is analyzed and interpreted. The interpretation is even more important in human-centered applications because of complexity of the human mental state. In this study, by big data challenge we refer to volume and velocity and we try to apply big data techniques in order to address this technology-side challenge of the problem. But utilizing big data techniques is a challenge itself that needs to be addressed first. Big data techniques are usually applied to vector space model. For example, for document similarity problem, each document is represented by a vector and each dimension shows the presence (binary vector) or the frequency (non-binary vector) of a term in the document. Also, there are similarity metrics used in these studies are usually designed to be applied to numeric vectors.

However, the application domain illustrated in this study is different. In this study, we have users with complex mental state surrounded by a set of sensors. We utilize Situ-Morphism to map *Situ* space to vector space first and then we show that similarity among situation sessions is preserved during this mapping.

# CHAPTER 6. LEARNING EXAMPLE

For better understanding the concepts, we adopt a simple smart home scenario designed in [12], [14] to illustrate the concepts used in this study. We extend some parts of the scenario based on the new definitions provided here. In this scenario, the behavior of an elderly w.r.t desire "taking a shower" has been observed for 10 days and results are shown in Table 6.1.

Table 6.1 A learning example showing situation sessions w.r.t. desire showering for 10 days

| ID:freq | Observed Paths |
|---|---|
| A:5 | grab towel → turn on the water → adjust the temperature of water → use soap → rinse body → turn off the water → drying body with towel → put some clothes on |
| B:1 | grab towel → turn on the water → open the fridge → grab food → close the fridge → Microwave the food → open the microwave → close the microwave → eat → adjust the temperature of water → use soap → rinse body → turn off the water → drying body with towel → put some clothes on |
| C:1 | grab towel → turn on the water → open the fridge → grab food → close the fridge → microwave the food → open the microwave → close the microwave → eat → brush teeth → use restroom → adjust the temperature of water → use soap → rinse body → turn off the water → drying body with towel → put some clothes on |
| D:3 | grab towel → turn on the water → use restroom → brush teeth → adjust the temperature of water → use soap → rinse body → turn off the water → drying body with towel → put some clothes on |

To simplify the representation of situation data, we show a compact version of situation from (d, A, E) to only (A). For example, action "Grab the towel" actually refers to the situation with environmental context values such as 2010-11-04 00:09:50.209589 M003 ON" as date, time and sensor location and status, desire "taking a shower". Session A with highest frequency has eight situations that are temporally ordered. The desire for all situations is same - "taking shower". The arrow in any session shows a situation transition such as: "Grab towel → turn on the water", "adjust the temperature of water → use soap" or "turn off the water → put some clothes on". In situation session A, "grab towel" is the initial situation and goal can be defined as a set of abstract logical rules "if user puts clothes on, turn off

water and leave the room". The end situation in session A, "put some clothes on" is assumed to have the goal (i.e. set of predefined logical rules) satisfied.

We introduce a ranking technique called Scalable *Situ* Ranking to rank situations based on their importance and filter out the unimportant situations. We first create a *Situ* graph and then rank the situations using Scalable Situ Ranking. Figure 6.1 shows the *Situ* graph corresponding the given learning scenario.

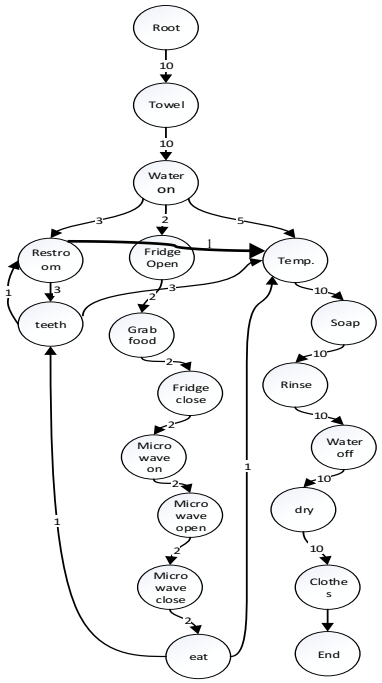



Figure 6.1 Situ graph corresponding the given learning example

The results of ranking situations from our example, are shown in Table 6.2. Here we consider any situation with rank higher than the average of ranks as milestone situation. Based on the application and the domain, the definition of milestone situation may vary. The threshold is specified to identify "higher" ranks and it is domain dependent and can be identified from different ways. In this thesis, we assume the average value of rank as the threshold to find milestone situations. Here, the average rank is 0.0535 that means any situation with rank greater

than or equal to 0.0535 is marked as milestone situation. Milestone situations are highlighted in the Table 6.2.

Table 6.2 Results of Situ ranking on given learning example

| Nodes | Rank | Nodes | Rank |
|---|---|---|---|
| grab towel | 0.0194 | close the fridge | 0.0144 |
| turn on the water | 0.0360 | microwave the food | 0.0161 |
| adjust the temperature of water | 0.1102 | open the microwave | 0.0176 |
| use soap | 0.1131 | close the microwave | 0.0188 |
| rinse body | 0.1161 | eat | 0.0199 |
| turn off the water | 0.1171 | brush teeth | 0.0280 |
| drying body with towel | 0.1181 | use restroom | 0.0231 |
| put on clothes | 0.1191 | | |
| open the fridge | 0.0100 | | |
| grab food | 0.0124 | | |

In this scenario, Scalable Situ Ranking is applied only on a subgraph and the situations pointing to initial situation "grab towel" have been discarded. Therefore, the value shown in Table 6.2 is lower than its actual Situ rank value since it is computed only based on the weight of situation and not the importance of links pointing to this situation.

After identifying milestone situations, we map *Situ* space to vector space using Situ-Morphism. To simplify representing situations, we assign a symbol to each situation. Table 6.3 shows the symbols defined for each situation.

Table 6.3 Situations and their corresponding symbols

| Situation | Symbol | Situation | Symbol |
|---|---|---|---|
| Grab towel | $S_1$ | adjust the temperature of water | $S_{10}$ |
| grab food | $S_2$ | eat | $S_{11}$ |
| brush teeth | $S_3$ | open the fridge | $S_{12}$ |
| use soap | $S_4$ | Rinse body | $S_{13}$ |
| Microwave the food | $S_5$ | open the microwave | $S_{14}$ |
| close the microwave | $S_6$ | close the fridge | $S_{15}$ |
| drying body with towel | $S_7$ | put some clothes on | $S_{16}$ |
| turn on the water | $S_8$ | turn off the water | $S_{17}$ |
| use restroom | $S_9$ | | |

In the given example, for two sessions A = $\{S_1 \rightarrow S_8 \rightarrow S_{10} \rightarrow S_4 \rightarrow S_{13} \rightarrow S_{17} \rightarrow S_7 \rightarrow S_{16}\}$ with size|A|=8 and C=$\{S_1 \rightarrow S_8 \rightarrow S_{12} \rightarrow S_2 \rightarrow S_{15} \rightarrow S_5 \rightarrow S_{14} \rightarrow S_6 \rightarrow S_{11} \rightarrow S_3 \rightarrow S_9 \rightarrow S_{10} \rightarrow S_4 \rightarrow S_{13} \rightarrow S_{17} \rightarrow S_7 \rightarrow S_{16}\}$ with size |C|=17, milestone situations w.r.t. desire="taking a shower" are MS = $\{S_1, S_4, S_7, S_8, S_{10}, S_{13}, S_{16}, S_{17}\}$. Milestone-position vector representation of situation sessions vectors of A and C can be represented as follows:

| | $S_1$ | $S_4$ | $S_7$ | $S_8$ | $S_{10}$ | $S_{13}$ | $S_{16}$ | $S_{17}$ |
|---|---|---|---|---|---|---|---|---|
| $A^{*\rightarrow}$= | <0.125, | 0.50, | 0.875, | 0.25, | 0.375, | 0.625, | 1, | 0.75> |
| $C^{*\rightarrow}$= | <0.058, | 0.764, | 0.941, | 0.117, | 0.705, | 0.823, | 1, | 0.882> |

Here, each dimension is a milestone situation and the ith entry shows the relative position of situation i in the corresponding session. For example, S1 is the first milestone situation appearing in A and length of situation A is 8. Therefore, the entry corresponding S1 in A* is 1/8 = 0.125.

In this learning example, $\mathrm{Sim_j}(A^{\rightarrow}, C^{\rightarrow}) = 1$ because all milestone situations appear in both sessions. Therefore, sessions A and C are similar Type I. Also, in given learning example, A′="$S_1S_8S_{10}S_4S_{13}S_{17}S_7S_{16}$", and C′ = "$S_1S_8S_{10}S_4S_{13}S_{17}S_7S_{16}$", $r_{A'}{}^{\rightarrow}$ and , $r_{C'}{}^{\rightarrow}$is a milestone order vector of sessions A and C, respectively. T = $\{S_1, S_4, S_7, S_8, S_{10}, S_{13}, S_{16}, S_{17}\}$ and $r_A{}^{\rightarrow} = < 0, 3, 6, 1, 2, 4, 7, 5 >$, $r_C{}^{\rightarrow} =< 0, 3, 6, 1, 2, 4, 7, 5 >$. Therefore $\mathrm{sim_o}(A^{\rightarrow}, C^{\rightarrow}) = 1 - \frac{||r_A{}^{\rightarrow} - r_C{}^{\rightarrow}||}{||r_A{}^{\rightarrow} + r_C{}^{\rightarrow}||} = 1 - \frac{0}{3.872474} = 1$ which means A and C are similar Type III with high degree (exactly same orders of milestone situations composition). Regarding similarity Type IV, $\mathrm{Sim_e}(A^{\rightarrow}, C^{\rightarrow})$ =1- 0.5124=0.487. Depending on the chosen threshold, we can consider A and C similar Type IV or not.

# CHAPTER 7. PROPOSED FRAMEWORK

After introducing Situ-Morphism, and showing that situation sessions can be compressed, we apply locality sensitive hashing technique sessions to find similar situation sessions, find nearest neighbors of a given session query or find groups of similar sessions. However, some situations such as “turning on TV" or “opening the fridge" w.r.t. desire “Sleeping” are not important enough to contribute much to the similarity of two sessions. Such situations are called non-milestone situations w.r.t desire d. Thus, the non-milestone situations are not features of any session. Therefore, we will first filter all non-milestone situations from every session using the Scalable Situ Ranking method discussed earlier in this thesis. We use the information of milestone situations (i.e. the order, position) as features of sessions. Then we use newly defined similarity metrics to measure similarity between situation sessions.

## Scalable Situ Ranking

In [12], we introduce a revised version of personalized PageRank called *Situ* Ranking in the context of situation-aware computing. In smart environments, the situation graphs are created from user’s situations as nodes and situation transitions as edges. The nodes and edges in situation graphs are labeled with meta information including situation elements (i.e. behavioral contexts, environmental contexts, user’s desire) and graph structure related information such as frequency of situation transitions. Therefore, *Situ* Ranking considers graph structure together with situation related meta information in order to assign higher ranks to the situations that matter more to achieve a specific goal. For this purpose, they bias the computation by extracting the hidden knowledge acquired from history of user’s

situations with respect to user's current desire. The formula for *Situ* Ranking is as follows:

$$SR^{k}(s_e) = \varepsilon \sum_{S_s \in In(S_e)} \left( SR^{k-1}(s_s) \times \frac{w_{S_s S_e}}{\sum_{S_z \in Out(S_s)} w_{sz}} \right) + (1-\varepsilon) \frac{w_{S_e}}{\sum_{S_j \in WS} w_{S_j}} \quad (8)$$ [12] ,[14].

Here WS is set of all situation nodes in *Situ* graph. An edge for *Situ* graph is represented as <id, $S_s$, $S_e$, $w_{se}$>. Here, id identifies an edge, $S_s$ shows the source situation and $S_e$ indicates the end situation corresponding to the link. $w_{S_s S_e}$ represents the weight of the edge connecting $S_s$ to $S_e$ and it is equals to the number of times situation $S_e$ has happened right after situation $S_s$. Here, $w_{S_i}$ represents the weight of situation $s_i$. The weight of a situation describes the number of times the user has been in that situation (i.e. $w_{S_e} = \sum_i w_{S_i e}$). The formula for *Situ* Ranking biases the PageRank calculation to assign higher ranks to the situations that were visited more often by the user in the past. Then, based on the rank of the situations, the ones with significantly higher ranks, called milestone situations, will be used for predictive analysis. In PageRank, the probability of jumping to another node is uniform but in *Situ ranking* it equals to the relative weight of the node. The weight of a situation corresponds to the frequency of the transitions from other situations to the situation. In other word, in *Situ* ranking all meta-information about the situations and situation transitions are considered in ranking process together with the graph topology.

Since situation graph includes all situation data associated with meta information about the graph structure, history of user's situations, implementing *Situ* ranking is computationally expensive in terms of time and space. In [14] we propose a scalable implementation of Situ Ranking using Map Reduce technique. Mapper is applied on every record of data in the form of (key, value) pairs and generates an arbitrary number of intermediate data records in the form of (key, value) pairs. Then reducer aggregates all intermediate pairs with same key and returns the output pairs [14]. We apply MapReduce in

two separate parts in order to compute the final rank of a situation at step k: computing the weight of situations at step k and computing situation ranks at step k.

**Computing Situations Weights at Step k**

The first part of formula for *Situ* ranking (i.e. $(1-\varepsilon)\frac{w_{S_e}}{\sum_{S_j \in WS} w_{S_j}}$) computes the relative weight of situation $S_e$. We define the weight of a situation as the number of transitions linking to the situation [12]. In other words, the weight of situation n is defined as $w_n = \sum_{m \in In(n)} w_{mn}$ where $w_{mn}$ is the frequency of transition from situation m to n. We propose the computation of situations' weight using MapReduce as follows:

**Map:** Take graph record <id, $S_s, S_e, w_{se}$> as input. Map the input on $S_e$ and emit <$S_e$, ($S_s, w_{se}$)> such that tuples with same $S_e$ are shuffled to the same machine. Such tuples are represented as <$S_e, \{S_s, w_{se}\}, S_s \in In(S_e)$>.

**Reduce:** Take <$S_e, \{S_s, w_{se}\}, S_s \in In(S_e)$> as input and emit <$S_e$, $w_{S_e}$> as output, where $w_{S_e}$ denotes the weight of situation node $S_e$ and its value is obtained by $\sum_{\forall S_s \in In(S_e)} w_{se}$.

In the mapper phase the data is mapped to the end situation, and the pairs are generated with the end situation as key and the start situation together with frequency of transition as value. Then all the pairs with same key are shuffled and sent to the reducer. The reducer sums up the frequency of transitions corresponding with the same key, and output the end situation as key and the weight of the situation as value. Figure 7.1 shows the pseudo code for computing situation weights in MapReduce.

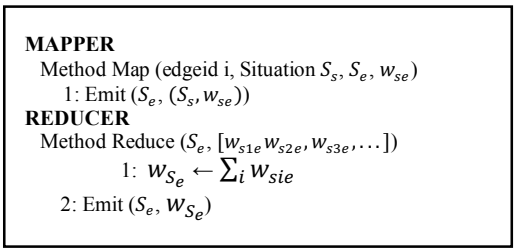



Figure 7.1 Pseudo-code for situation's weight computation in MapReduce

**Computing Situations Ranks at Step k**

In order to calculate *Situ* rank of a situation at step k, all the in-degree situations pass their rank computed at step k-1 to the current situation via outgoing transitions ($\varepsilon \sum_{S_s \in In(S_e)} \left( SR^{k-1}(s_s) \times \frac{w_{se}}{\sum_{S_z \in Out(S_s)} w_{sz}} \right)$). After each iteration, all ranks of situations computed at step k-1 linking to the current situation are summed up and the new rank of situation at step k is updated. The iterative process stops when the situations' rank values do not change anymore. We propose two levels of MapReduce in order to compute the second part of the formula [14]: First level implements $\sum_{S_z \in Out(S_s)} w_{sz}$ that indicates the sum of the weights of outgoing links from the situation linked to the current situation. In the second level, the ranks of in-degree situations are passed to outgoing situations and all the ranks at step k get updated. Mapper in level two iterates over the situation nodes, and distributes the current rank score (i.e. $\pi_s$) to its outgoing neighbors by generating pairs including the partial value of the rank as value and the id of the outgoing neighbors as key. All the values with same key are aggregated by summing up the rank value contributions from all in degree transitions and output the updated rank (i.e. $\pi'_e$) as new value and situation id as key. We propose to compute situations ranks using two levels of MapReduce as follows:

**Map I**: Take graph record <id, $S_s, S_e, w_{se}$ > as input. Map the input on $S_s$ and emit <$S_s$, ($S_e, w_{se}$)> such that tuples with same $S_s$ are shuffled to the same machine. <$S_s$, {$S_s, w_{se}$}, $S_e \in$ Out ($S_s$)>.

**Reduce I**: Take <$S_s$, {$S_s, w_{se}$}, $S_e \in$ Out ($S_s$)> and emit <$S_s$, $w_{Out_s}$> where $w_{Out_s}$ denotes the summation of frequency of all outgoing transitions from situation node $S_s$ and its value is obtained by $\sum_{\forall S_e \in Out\,(S_s)} w_{se}$.

**Map II**: Take the record <id, $S_s, S_e, \pi_s, w_{se}, w_{Out_s}$> as input. Map the input on $S_e$ and emit <$S_e$, ($S_s, \pi_s * \frac{w_{se}}{w_{Out_s}}$)> such that tuples with same $S_e$ are shuffled to the same machine. <$S_e$, {$\pi_s * \frac{w_{se}}{w_{Out_s}}$}, $S_s \in$ In($S_e$)>.

**Reduce II**: Take <$S_e$, {$\pi_s * \frac{w_{se}}{w_{Out_s}}$}, $S_s \in$ In ($S_e$)> and emit <$S_e$, $\pi'_e$> where $\pi'_e = \sum_{\forall S_s \in In\,(S_e)} \pi_s * \frac{w_{se}}{w_{Out_s}}$.

Figure 7.2 and Figure 7.3 represent the pseudo code for the map and reduce tasks in both levels I and II.

```
MAPPER I
  Method Map (id, S_s, S_e, w_se )
    1: Emit <S_s, (S_e, w_se)>
REDUCER I
  Method Reduce (S_s, [w_se1, w_se2, . . .])
    1: for all w_sei ∈ [w_se1, w_se2, . . .] do
      2: w_Out_s ← w_Out_s + w_sei
    3: Emit (S_s, w_Out_s)
```

Figure 7.2 Pseudo-code for Situ ranking in MapReduce (I)

**MAPPER II**

Method Map (id, $S_s, S_e, \pi_s, w_{se}, w_{Out_s}$)

1: $\pi'_e \leftarrow \pi_s * \frac{w_{se}}{w_{Out_s}}$

2: Emit <$S_e$, ($S_s, \pi'_e$)>

**REDUCER II**

Method Reduce ($S_e$, [$\pi'_{1e}, \pi'_{2e}$. . .])

1: for all $\pi'_{ie} \in [\pi'_{1e}, \pi'_{2e}$. . .] do

2: $\pi'_e \leftarrow \pi'_e + \pi'_{ie}$

3: Emit ($S_e$, $\pi'_e$)

Figure 7.3 Pseudo-code for situ ranking in MapReduce (II)

In Figure 7.5 and Figure 7.6, we illustrate running of an iteration of situation ranking computation in MapReduce on a sub graph depicted in Figure 7.4.

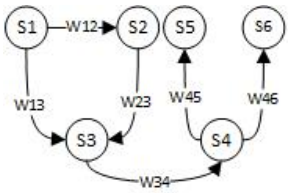



Figure 7.4 A small subset of situation graph

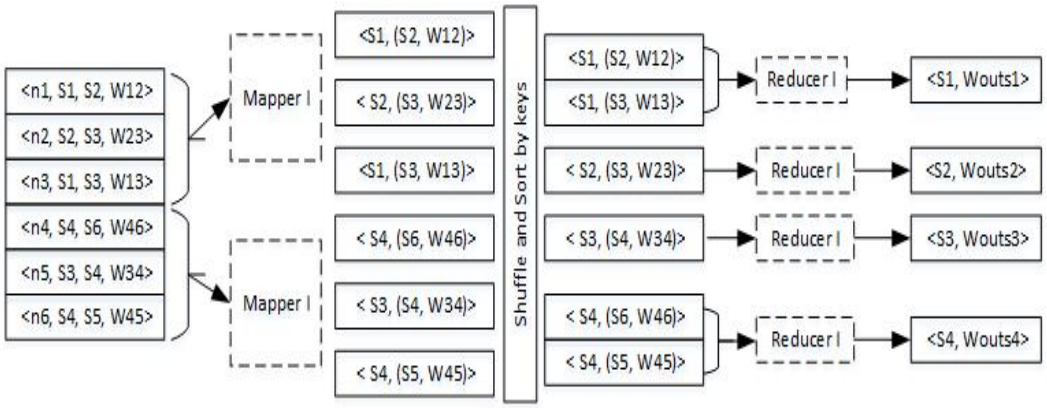



Figure 7.5 An example of an iteration of situation ranking in MapReduce (I)

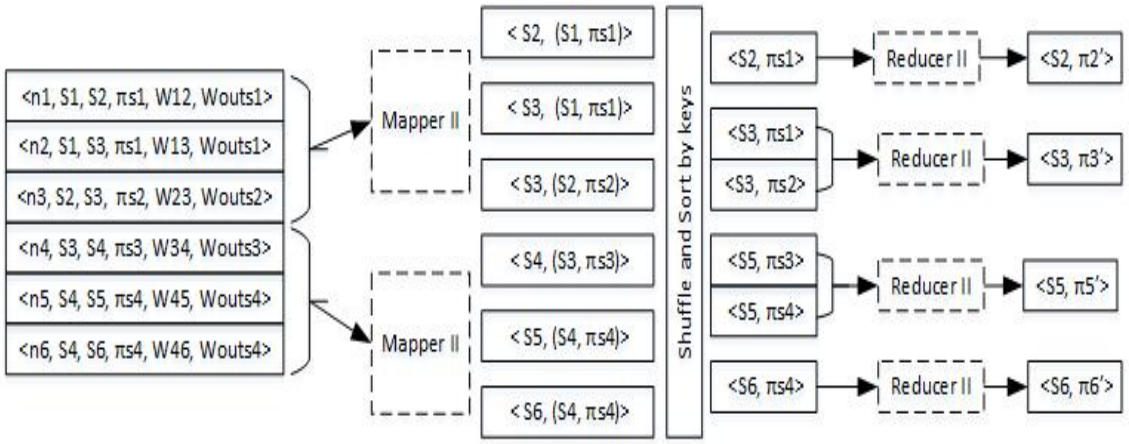



Figure 7.6 An example of an iteration of situation ranking in MapReduce (II)

**Creating Probabilistic Graphical Model**

We build a probabilistic graphical model combined with Situ ranking technique to predict the user's next situation and then we use this as a base model to compare with approximate approaches for prediction. We utilize a higher order Markov model in order to predict the resident's future situations and make the right decision accordingly [12]. Higher-order Markov models provide more effective predictions compared to lower-order Markov models. The reason is that lower-order models do not take the past into account in order to predict user's future. Keeping a sequence of a user's situations allows the model to correctly distinguish the differences between observed patterns and make a reliable prediction [12],[14]. We build the predictive model based on milestone situations and milestone transitions. After the model is built based on training data, it can be used for predictive analysis. The model predicts the resident's future situations by finding the most similar situation path to the partially observed path and makes the right decision accordingly.

## Approximate Nearest Neighbor Search

### Estimating Similarity Type II

To estimate similarity type II between two sessions first we assume MS = {1, 2, …, m} is the set of milestone situations that appear in union two sessions. Let $MS_a$ be the set of milestone situations appearing in session $Ses_a$ and let $MS_b$ be the set of milestone situations that appear in $Ses_b$. Using LSH we can estimate this similarity in a more efficient way:

A permutation on MS is a one-one, onto function from MS to MS. For a random permutation II on MS, Pr[min[Π($Ses_a$)] = min[Π($Ses_b$)]] = Similarity Type II [97][86]. In summary, the steps to estimate similarity Type II are shown below:

*Input:* $Ses_a$, $Ses_b$ *and MS set of all milestone situations in* $Ses_a \cup Ses_b$

1. *Uniformly at random pick k permutations* $\Pi_1, \ldots, \Pi_k$
2. *Set* $MH_a = <min[\Pi_1(Ses_a)]\ min[\Pi 2(Ses_a)], \ldots, min[\Pi k(Ses_a)]>$
3. *Set* $MH_b = <min[\Pi_1(Ses_b)]\ min[\Pi 2(Ses_b)], \ldots, min[\Pi k(Ses_b)]>$
4. *Set l as the number of items where* $MH_a$ *and* $MH_b$ *match, i.e.* $l = |\{i | min[\pi_i(Ses_b)] = min[\pi_i(Ses_a)],\ 1 \leq i \leq k\}|$
5. *return* $\frac{l}{k}$

Figure 7.7 The algorithm to estimate similarity Type II

Using Chernoff bounds it is shown that given $k = O(\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$, we have:

$$\Pr\left[\left|\text{Similarity Type II}(Ses_a, Ses_b) - \frac{l}{k}\right| \leq \epsilon\right] \geq 1 - \delta \quad (9).$$

Therefore, two vectors MHa and MHb contain sufficient information to estimate similarity Type II of $Ses_a$ and $Ses_b$.

In computing exact similarity Type II($Ses_a$, $Ses_b$) we use $MS_a$ and $MS_b$, where $MS_a$ and $MS_b$ are m-dimensional vectors. Compared to that MHa and MHb are k-dimensional vectors and k << m. Thus, we have “compressed" the entire situation session A to a bunch of k-numbers, and these numbers (along with k permutations) are sufficient to estimate the

similarity of $Ses_a$ and any other sessions. The vector MHa is a signature of the session $Ses_a$. Since the technique used to obtain this signature is the minimum value of (one-one) hash functions, this is called minhash signature [98][89].

For collection of situation sessions Ses = $\{Ses_1,..., Ses_N\}$ and milestone situations MS = $\{ms_1,..,ms_m\}$ appearing in the collection Ses, we assign the milestone situations to the set of integers: {1,…,M}. The milestone situation-session matrix is the following MxN matrix, where $b_{ij}$ equals 1 if the situation $mS_i$ appears in session $Ses_j$, otherwise $b_{ij}$ equals 0.

| | $Ses_1$ | …. | $Ses_N$ |
|---|---|---|---|
| $ms_1$ | $b_{11}$ | …. | $b_{1N}$ |
| $ms_2$ | $b_{21}$ | …. | $b_{2N}$ |
| . | . | …. | . |
| $ms_m$ | $b_{m1}$ | …. | $b_{mn}$ |

The MinHash matrix of the above collection is a kxN matrix obtained by randomly picking k permutations $\{\pi_1,…, \pi_k\}$ each permutation is from {1,…,m} to {1,...,m}.

| | $Ses_1$ | …. | $Ses_N$ |
|---|---|---|---|
| $\pi_1$ | $\min[\pi_1(Ses_1)]$ | …. | $\min[\pi_1(Ses_N)]$ |
| $\pi_2$ | $\min[\pi_2(Ses_1)]$ | …. | $\min[\pi_2(Ses_N)]$ |
| . | . | …. | . |
| . | . | …. | . |
| $\pi_k$ | $\min[\pi_k(Ses_1)]$ | …. | $\min[\pi_k(Ses_N)]$ |

In general, the ijth entry of the MinHash matrix is $\min[\pi_i(Ses_j)]$. The ith column of Minhash Matrix shows the MinHash Signature of situation session $Ses_i$: < $\min[\pi_1(Ses_j(Ses_i)]$,… $\min[\pi_k(Ses_j(Ses_i)]$>.

We utilize locality sensitive hashing technique in order map similar sessions into the same bucket. Let M be the k × M MinHash matrix. We partition rows of M into b bands such that each band has r rows (thus k = rb). For a session $Ses_i$, its MinHash signature is $MH_i = < n_1, \dots, n_k >$. The entries of MHi appear in the ith column of M. We annotate the first entries of $MH_i$ with $MH_i^1$, second entries with $MH_i^2$ and so on. Thus, $MH_i^l$ is a r-tuple. Randomly we

pick a hash function h from a set of r-tuples to {1,…,T} (where T > N). T1, T2,…,Tb are b hash tables. More precisely each Ti is an array of size T and each cell of the array points to a list. To identify candidate similar pairs for every i ∈ {1,…, N} we compute $h(MH_i^1)$, $h(MH_i^2)$,…, $h(MH_i^b)$ and for every $l \in \{1, \dots, b\}$ place $Ses_i$ in the list at $T_l[h(MH_i^l)]$. Suppose that two sessions $Ses_i$ and $Ses_j$ are s-similar, i.e., Similarity Type II ($Ses_i$, $Ses_j$) = s. The probability that both $Ses_i$ and $Ses_j$ are mapped to the same bucket in some hash table is $s^r$. Given a hash table $T_l$ $Ses_i$ and $Ses_j$ are placed into the same bucket of $T_l$ if $h(MH_i^l)$ equals $h(MH_j^l)$. Since $Ses_i$ and $Ses_j$ are s-similar, the probability that any two corresponding milestone situations of MHi and MHj are the same is s. Thus, the probability that $MH_i^l$ and $MH_j^l$ are same is at least $s^r$. In other words, the probability that $Ses_i$ and $Ses_j$ are placed in different buckets of $T_l$ is at most $(1 - s^r)$. Similarly, the probability that $Ses_i$ and $Ses_j$ are placed in different buckets of every hash table is at most $(1 - s^r)^b$. And the probability that $Ses_i$ and $Ses_j$ are placed into the same bucket of some hash table is at least $1 - (1 - s^r)^b$.

To summarize, below is the algorithm to identify near duplicate sessions in a collection of sessions w.r.t similarity Type II:

*Input: Ses = {$Ses_1, \dots, Ses_N$}, Set of all situations S = {$s_1, \dots, s_m$}, Query session $Ses_q$*

1. *Apply Scalable Situ Ranking to identify milestone situations*
2. *Apply Situ-Morphism to map Situ space to vector space*
3. *Assigning each milestone situation with an integer so that MS is seen as {1,..,M}*
4. *Uniformly at random pick k permutations Π1,…,Πk from {1,…,M} to {1,..,M}.*
5. *Compute the MinHash Matrix M.*
6. *Partition rows of M into b bands with each band having r-rows.*
7. *Pick a hash function h and b hash tables T1,…, Tb.*
8. *Hash the situation sessions into hash tables T1,..,Tb.*
9. *Identify sessions that are mapped into the same bucket as a group of similar sessions.*
10. *Return sessions in the bucket that includes query session $Ses_q$*

Figure 7.8 The algorithm to identify near duplicate sessions in a collection of sessions w.r.t similarity Type II

**Estimating Similarity Type III**

There is no known LSH for this order similarity measure but since the formula for similarity Type III includes length of $||\mathrm{r_A}^{\rightarrow} - \mathrm{r_C}^{\rightarrow}||$ and $||\mathrm{r_A}^{\rightarrow} - (-\mathrm{r_C}^{\rightarrow})||$ so we can use the same LSH proposed for Euclidean distance to estimate this measure as well.

**Estimating Similarity Type IV**

The hash functions here correspond to lines and lines are partitioned into buckets of size a. Given a situation session, the hash function maps it to the bucket containing its projection onto the line. Therefore, the sessions that are similar w.r.t. similarity Type IV are likely to put into same bucket. Formally, if (1- similarity Type IV) >> a, the angle between two sessions should be close to 90 degrees for there to be any chance they go to same bucket. If (1- similarity Type IV) << a, then the chance the sessions are in same bucket is at least 1-(1- similarity Type IV)/a. For any two situation sessions $\mathrm{Ses_a}$ and $\mathrm{Ses_b}$, if (1 - similarity Type IV($\mathrm{Ses_a}, \mathrm{Ses_b}$) ) $\geq$ 2a then the angle $\theta$ should be between 60 and 90 for there to be a chance that these sessions be considered similar Type IV and go to same bucket. However, if (1 - similarity Type IV($\mathrm{Ses_a}, \mathrm{Ses_b}$) ) $\leq$ a/2 then there is at least ½ chance that they be considered similar Type IV. And this yields a (a/2, 2a, 1/2, 1/3)-sensitive family of hash functions to estimate similarity Type IV of situation sessions [83].

*Input: Ses = {$Ses_1$,..., $Ses_N$}, Set of all situations S = {$s_1, \dots, s_m$}, Query session $Ses_q$*

1. *Apply Scalable Situ Ranking to identify milestone situations*
2. *Apply Situ-Morphism to map Situ space to vector space*
3. *Assigning each milestone situation with an integer so that MS is seen as {1,..,M}*
4. *Take a random unit vector u∈ $R^d$. A unit vector u satisfies that ||u||=1, that is d(u,0) = 1*
5. *Project two sessions $Ses_a$, $Ses_b$ onto u: $Ses_{a_u} = < Ses_a, u > = \sum_{i=1}^{k} a_i.u_i$ and this is contractive so $||Ses_{a_u} - Ses_{b_u}|| <= ||Ses_a - Ses_b||$*
6. *Create buckets of size a on u (in $R^1$). The index of bucket $Ses_a$ falls into h($Ses_a$).*
7. *If $||Ses_a - Ses_b|| < a/2$ then Pr[h($Ses_a$)= h($Ses_b$)]$\geq 1/2$.*
8. *If $||Ses_a - Ses_b|| > 2a$ then Pr[h($Ses_a$)= h($Ses_b$)]$< 2/3$*

Figure 7.9 The algorithm to estimate similarity Type IV of situation sessions

# CHAPTER 8. CASE STUDIES

## Dataset Creation

The cost to create real smart homes and the collection of datasets is expensive and sometimes infeasible for many projects because of issues such as finding the optimal location to deploy sensors, lack of flexibility, finding participants and privacy and ethical concerns. There are some real smart home datasets available for research. However, oftentimes, they do not meet the needs of the conducted research projects. In our case, these open datasets fail to support our need to add desire labels to sensor events or an appropriate annotation method for the inhabitants' activities. As open-source data were collected already, there is no way to go back and add some sensors, or change the design of the smart home. To overcome the drawbacks of generating real datasets we take advantage of smart home dataset simulation tools.

There exist some studies proposing smart home simulation tools. However, not all of them are open-source and some of them lack the flexibility to customize the smart home design by adding and/or removing sensors and smart devices. There are two approaches to generate datasets: model-based approaches and interactive approaches. Model-based approaches are capable of generating bigger datasets, but with fewer interactions in real time. However, the interactive approaches usually produce the datasets in a longer time but with more interactions in real time.

To generate data for our research, we use the simulator proposed in [13] called OpenSHS. OpenSHS is a hybrid, open-source, cross-platform 3D smart home simulator. The pleasant feature of this simulator is scalability which means that the sample dataset produced, can be extended without affecting the logical order of the events to help with generating big

smart home datasets. OpenSHS combines advantages from both interactive and model-based approaches with reducing the time and effort required to generate simulated smart home datasets while providing an interaction scheme between the simulated smart home and simulated participants in real time. OpenSHS tool is based on Blender and Python, which are both open-source and it works on multiple platforms.

In this phase, as shown in Figure 8.1, we build the virtual environment, import the smart devices, assign labels to activities and design the contexts.

Figure 8.1 An example of a designed smart home using Blender [13]

Using Blender, we designed the 3D smart home consisting of a bedroom, a bathroom, a living room, a kitchen and an office. Each room is equipped with several sensors. There are twenty-nine sensors of different types in total. To simplify the case and reduce the amount of data generated, the sensors are taken to be binary, and they are either on or off at any given time step. Then smart devices such as TV, oven, fridge, etc. are imported into the smart home from the smart devices library, offered by OpenSHS. The smart devices library includes different types of smart devices and sensors and it is extensible. The sensors that we import to our smart home are as follows: pressure sensors placed under carpet, bed and couch, door sensors, lock devices, appliance switches such as TV, oven, fridge, and light controllers. OpenSHS enables the researchers to define an unlimited number of desires. We identify 10

labels, namely, ‘sleeping’, ‘cooking’, ‘bathing’, ‘leaving’, ‘studying’, ‘entering’, ‘eating’, ‘watching TV’, ‘anomaly’, ‘other’. Figure 8.2 shows the desires appearing on top right of the screen so user can choose the desire and start the new session. Also, we can define specific time frames as environmental contexts, e.g., morning, afternoon, evening, weekdays, weekend afternoon, etc. We define four time frames: weekday morning, weekday evening, weekend morning, weekend noon, weekend evening. Each context has a default starting date and time, and the researcher can adjust the date and time as she wants to. Every context has an initial state for the sensors and the 3D position of the participant (as mentioned earlier, we only consider one user for this study). The default sample rate in OpenSHS is one second but it can be re-configured to finer grained as required. Another interesting feature of OpenSHS is fast-forwarding. It allows the user to control the time span of a certain activity. For example, in Figure 8.3 the participant wants to watch TV for a period of time and does not want to perform the whole activity in real time, the user can initiate the activity and specify how long the activity lasts. The tool will simply copy and repeat the existing state of all sensors and devices during the given time period.

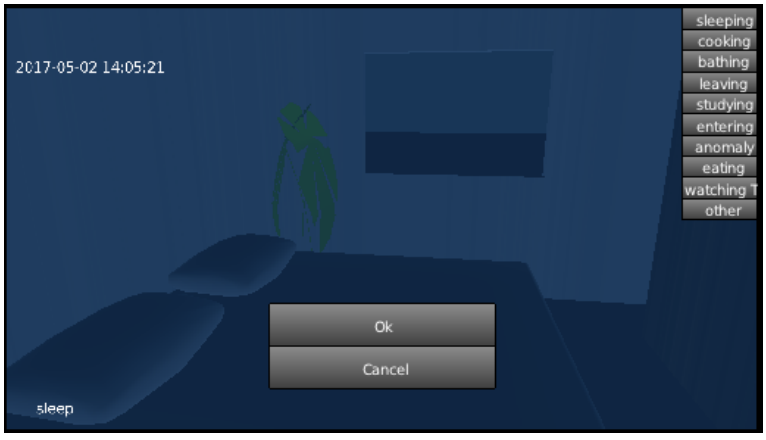



Figure 8.2 Customized list of desires appearing on screen


Figure 8.3 Activity fast-forwarding dialogue during a simulation

After performing simulation by the user, we aggregate the generated sample activities for each time frame, in order to produce the final dataset. In order to generate big data, OpenSHS propose a technique to replicate the recorded samples without affecting their logical order. During aggregation and generation of the final dataset, the samples of every context are grouped by the number of activities in each sample. Then, a random group will be chosen, and from that group, a sample will be drawn for each activity. In general, the number of uniquely replicated copies for a single context can be calculated by $R = \sum_{g=1}^{G} S_g^A$. G denotes the number of the groups of unique length of activities and Sg denotes the number of samples for the group g and A denotes the number of activities within a sample Sg.

Figure 8.4 shows the Blender interface for context monitoring and Figure 8.5 displays an example of interaction between smart home and participant. Using OpenSHS we are able to customize the initial state of different contexts. Figure 8.6 shows an example of initial context values. Figure 8.7 lists some of the generated datasets for different time frames and Figure 8.8 is an example of content of generated data. Starter desire is “leaving” and interleaving desire is “bathing”. Each row shows a situation containing three elements of user’s starter desire, environmental contexts (here, time and location) and behavioral context

(changes in sensor events inferred as an action). We can show the above sensor log as two situation sessions:

1. Open wardrobe → close wardrobe → leave bedroom → close bedroom door → enter bathroom → turn on bathroom light → turn off bathroom light → leave bathroom → unlock main door → open main door → close main door → lock main door
2. Enter bathroom → turn on bathroom light → turn off bathroom light → leave bathroom

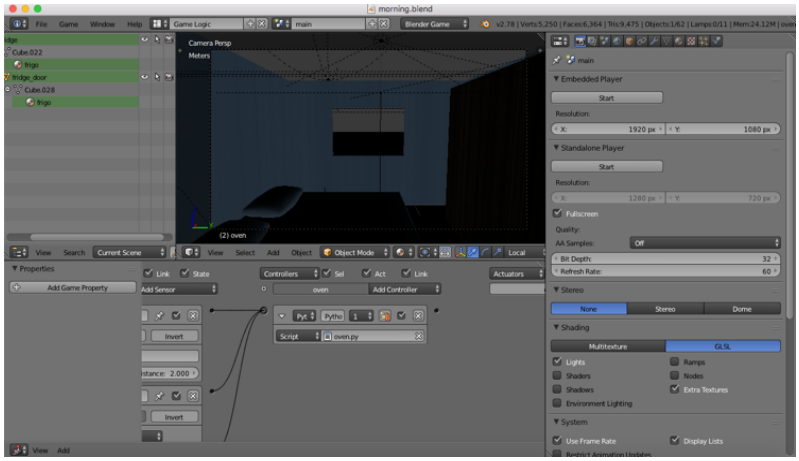


Figure 8.4 Blender interface for context morning


Figure 8.5 An example of interaction between smart home and participant

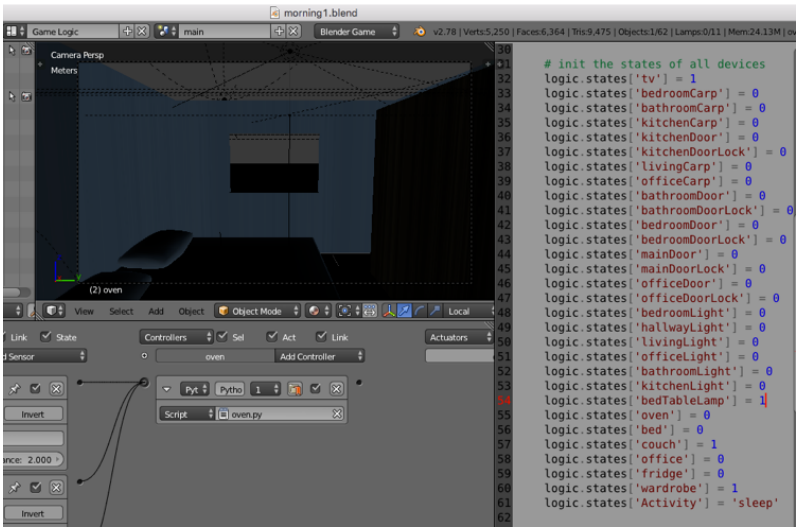
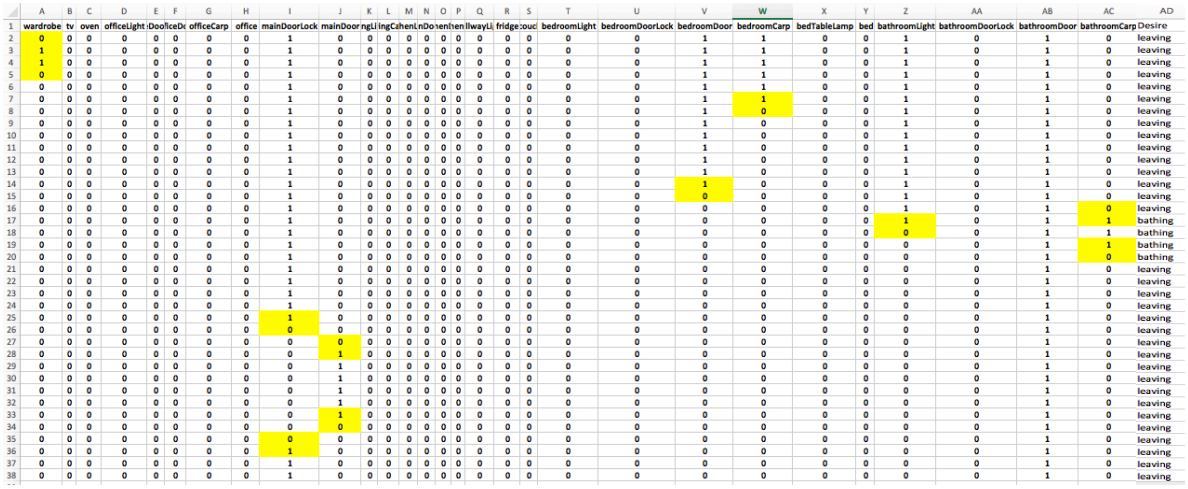



Figure 8.6 Customizing initial state of different contexts


Figure 8.7 An example of generated datasets for different time frames



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | AA | AB | AC | AD |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | wardrobe | tv | oven | officeLight | DoorDe | officeDe | officeCarp | office | mainDoorLock | mainDoor | ngLi | ingC | ehenLi | nDo | eni | nen | llwayLi | fridge | ouc | bedroomLight | bedroomDoorLock | bedroomDoor | bedroomCarp | bedTableLamp | bed | bathroomLight | bathroomDoorLock | bathroomDoor | bathroomCarp | Desire |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | leaving |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | bathing |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | bathing |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | bathing |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | bathing |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 31 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 32 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 34 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 35 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 36 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 37 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |
| 38 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | leaving |

Figure 8.8 A sample of final dataset output

# CHAPTER 9. STUDY RESULTS

We've evaluated different components of the proposed framework separately. First, we evaluate *Situ* ranking on top of a predictive probabilistic graphical model in comparison with a base probabilistic graphical model with no ranking. Then we evaluate scalability of our work by increasing the data and testing the run time for prediction. Also, we evaluate approximate nearest neighbor search on a simulated dataset to compare the memory consumption and run time compare to the methods that utilize exact comparisons. The implementation used for our evaluation is programmed in Python.

## Probabilistic Graphical Model with Situ Ranking

To evaluate *Situ* Ranking, we employed one data set called Aruba [99]. The data set includes sensor data collected in a smart home with one resident and three different sensors: motion sensors, door closure sensors and temperature sensors. The sensor events are labeled with user's desire that is a desired property of this data set to be used in this study. There are eleven desire labels including "meal preparation" with 1606 situation sessions, "relax" with 2910 situation sessions, "eating" with 257 situation sessions, "work" with 171 situation sessions, "sleeping" with 401 situation sessions, "wash dishes" with 65 situation sessions, "bed to toilet" with 157 situation sessions, "enter home" with 431 situation sessions, "leave home" with 431 situation sessions, "house keeping" with 33 situation sessions, and "respiration" with 6 situation sessions. The layout of the sensors embedded in the smart home is shown in Figure 9.1. The dark points indicate the sensors.

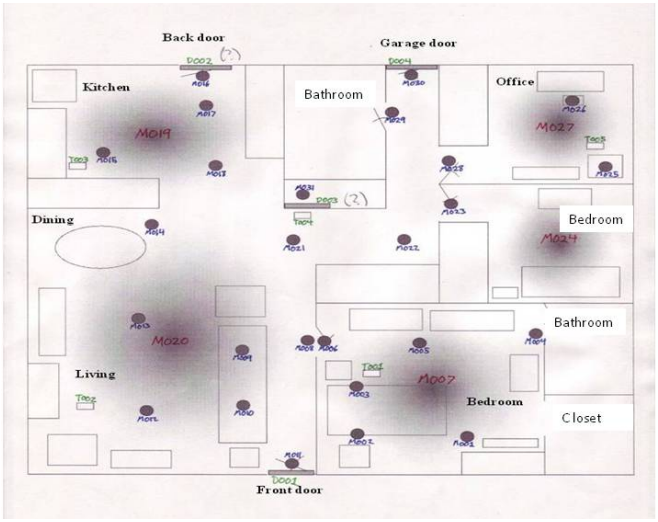



Figure 9.1 The layout of the sensors embedded in the smart home [99]

Aruba dataset is approximately 61 MB that contains raw sensor events collected in 2010–2011. The situations are recorded in terms of sensor status (3 types of sensors: motion, door, temperature) annotated with the desire of user (11 categories). We evaluated this approach in three different setups of the prediction model. The first setup is pure Markov model and the second one is Markov model extended with ranking based on PageRank. The third setup is the one with *Situ* ranking. As the reader can observe from Figure 9.2, the proposed approach provides an improvement over the other two setups in terms of accuracy of prediction[2]. The reason is that pure Markov model assigns equal prior probabilities to all nodes but in the two other setups, Markov model is extended with ranking process which uses the ranks as prior probabilities. Between PageRank and *Situ* ranking, *Situ* ranking provides more objective and accurate predictions. Figure 9.3 depicts the OSim[3] similarity for the top 3, 5, 8 rankings of Aruba dataset w.r.t. d=" Meal-Preparation". The top-n rankings

[2] Precision = $\frac{\text{Number of correct predictions retrieved (TP)}}{\text{Number of correct predictions retrieved (TP)+Number of wrong predictions retrieved (FP)}}$

[3] OSim($\tau1$, $\tau2$) indicates the degree of overlap between the top n situations of two rankings, $\tau1$ and $\tau2$. OSim($\tau1$, $\tau2$)=$\frac{|\tau1 \cap \tau2|}{n}$ [76].

from *Situ* ranking are more similar to actual user's behavior compared to PageRank based ranking. Figure 9.4 shows the comparison of three setups in terms of complexity. However, the higher order Markov model with *Situ* ranking utilizes look far into the past in order to provide more accurate user's situation and next path prediction, it only considers milestone situations and transitions to compress the model. As a result, as Figure 9.4 shows, the predictive model with *Situ* ranking has lower complexity and prediction time compare to two other setups.

In Table 4.1, we compare proposed predictive model with *Situ* ranking [14] with some other probabilistic graphical model-based approaches [71], [72], [100] in terms of prediction accuracy. Please note that the dataset used in [100] was not available and the datasets used in [71][72] were not annotated with the user's desires so they were not applicable to this study. Nevertheless, we employed Aruba dataset to evaluate Situ ranking component of our research. According to the authors of [100], Aruba is closer to dataset used in [100] with same assumptions and features compared to other available smart home datasets. Also, Aruba dataset includes user's activity data collected for two years; however, the dataset used in [71], includes user's activity data for only two months. Considering the fact that Aruba has much more data than dataset used in [71], our approach has comparable (i.e. against SPEED [71] that was validated for only two-month data) accuracy while keeping the complexity low which is an improvement over that of [71], [72], [100]. We speculate the reason that predictive model combined with *Situ* ranking outperforms the other three approaches is because it ranks the situations dynamically based on the user's current desire. Also, it compresses the representation model by identifying milestone situations and milestone transitions.

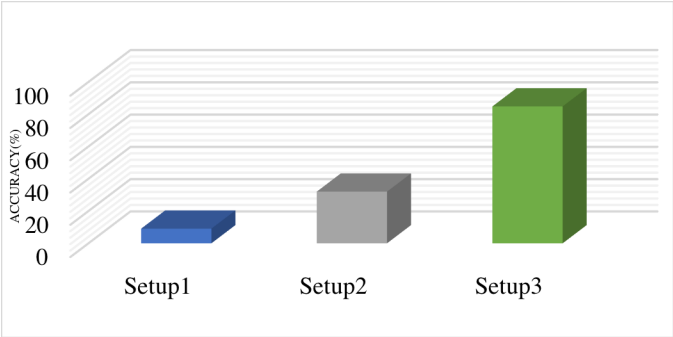



Figure 9.2 Comparison of setup1 (pure Markov model), setup2 (Markov model extended with ranking) and setup3 (Markov model extended with Situ ranking) in terms of accuracy of prediction

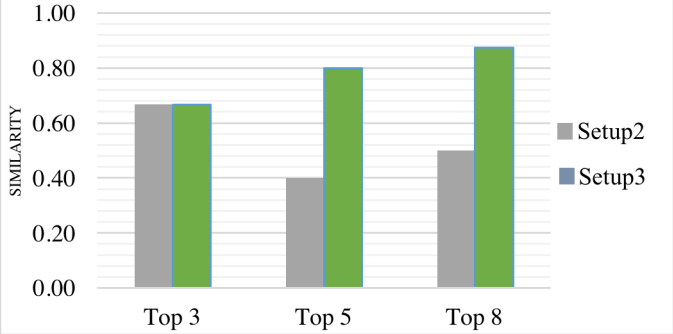



Figure 9.3 Comparison of setup2 (Markov model extended with ranking) and setup3 (Markov model extended with Situ ranking) in terms of similarity of top-n (n=3, 5, 8) rankings to the user's actual data w.r.t. desire=Meal-Preparation


Figure 9.4 Comparison of setup1 (pure Markov model), setup2 (Markov model extended with ranking) and setup3 (Markov model extended with Situ ranking) in terms of prediction time.

Table 9.1 Comparison of predictive graphical model combined with Situ ranking with some other predictive graphical model-based approaches

| **Name** | **Method** | **Accuracy** |
|---|---|---|
| *CRAFFT* [100] | *Dynamic Bayesian Network* | *74.75%* |
| [72] | *Markov Logic Networks* | *65%* |
| *SPEED* [71] | *Markov models with partial matching* | *88.3%* |
| *Higher Order Markov Model with Situ Ranking* [14] | *Higher order Markov models extended with Situ ranking* | *84.84%* |

**Scalable Situ Ranking**

To evaluate scalability of *Situ* ranking, we did some preprocessing to annotate the raw sensor events with situation data and create intermediate files such as a file containing situation nodes, a file for situation transitions, one for each desire, etc. All these processing extended the size of the whole Aruba dataset to approximately 2 GB. We use the extended Aruba dataset to evaluate the scalability of the proposed ranking technique. All our

experiments were performed on an experimental cluster of three machines each with 2.5 GHz with 1Gb main memory. To test the scalability of the proposed ranking technique, we divide the dataset into subsets of 10, 30, 50, 70, 90, and 100 percent of the input data and run the algorithm for each subset. Larger datasets collected from a longer period are certainly desirable; however, they are in general hard to come by. Note that the WSU[4] open datasets are among the most popular ones adopted by the smart health community and contain data meeting our need. Here, we use the same predictive model combined with Situ ranking in previous section. The only difference is that, we implement the Situ ranking using MapReduce to make the ranking process scalable. We design an experiment to analyze the time complexity of serial [14] and parallel implementation (i.e. proposed MapReduce technique) of our situation ranking algorithm. Figure 9.5 demonstrates the experimental results of the comparison in terms of run time. The parallel implementation of situation ranking algorithm by MapReduce can effectively process large-scale situation-annotated sensor data and accelerate the algorithm. The reason is that in serial algorithms, a problem is broken into a series of instructions and then the instructions are executed sequentially one after another. However, in parallel algorithms, a problem is broken into discrete sub problems that can be solved concurrently. Then, each sub problem is broken down to a series of instructions that are executed simultaneously on different processors [101]. The results of our experiments show that given a dataset the parallel algorithm for *Situ* ranking is roughly three times faster than its serial version. Figure 9.6 shows the scalability of the proposed algorithm for ranking situations. As observed in Figure 9.6, the increase in time is almost linear in the size of the input data that is a desirable property.

[4] Washington State University

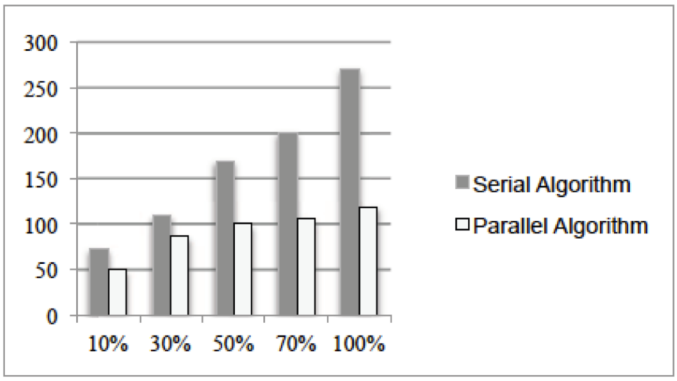



Figure 9.5 A comparison between run time of Situ ranking Algorithm in two parallel and serial mode with scaling data

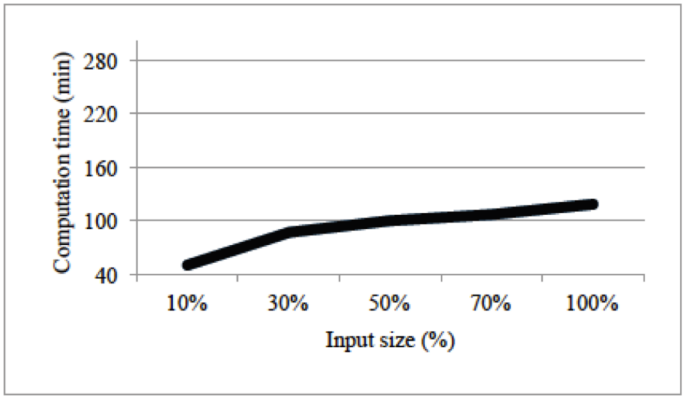



Figure 9.6 Run time of Situ ranking for subsets of Aruba dataset

In terms of space scalability, the MapReduce algorithm generates intermediate data in order of number of edges. And because situation graph is a sparse graph, the number of edges is in order of number of nodes. Therefore, the space will increase linearly when the input size increases. Also, the results of the experiments for accuracy of the predictions show that the accuracy remains the same as the one reported in [14] which is approximately 84.84%.

## Approximate Nearest Neighbor Search with Situ-Morphism

As probabilistic graphical models are expensive to build and use when the data is massive, we utilize approximate pattern matching approach for prediction using the novel concept Situ-Morphism. We generate ~7GB data by simulating a smart home using OpenSHS in about two weeks in a relatively realistic setting. The data includes about 1 million sessions (samples) with 30,000 situations (dimensions). After mapping Situ space to vector space using Situ-Morphism, the output is a 1,000,000 x 30,000 sparse matrix which takes a lot of space. To address this challenge, we apply Scalable Situ Ranking [14], and we utilize a revised version of the existing implementation of LSH to find near neighbor sessions of test query sessions. We use an AWS Linux instance with 16GB main memory and 4 vCPUs. We evaluate the performance of our framework in terms of memory consumption, accuracy and query speed compared to our benchmark models for predictive analysis: 1) Naïve predictive model that uses brute force similarity search with no *Situ* ranking, 2) predictive model that only applies *Situ* ranking and 3) our proposed predictive model.

### Memory Consumption

To evaluate memory consumption, we compare three approaches for predictive analysis. The first one is the brute force case that considers all situations as dimensions of situation vectors and does a pair similarity search to find similar sessions to user's current session and provide predictions accordingly. The second approach is an improved version of first approach and it reduces the dimensions to milestone situations but still it uses a naïve pair similarity search to find similar sessions and use them for prediction. The third approach is the proposed framework in this study. LSH helps with dimension reduction as well. Memory consumption for each approach has shown below.

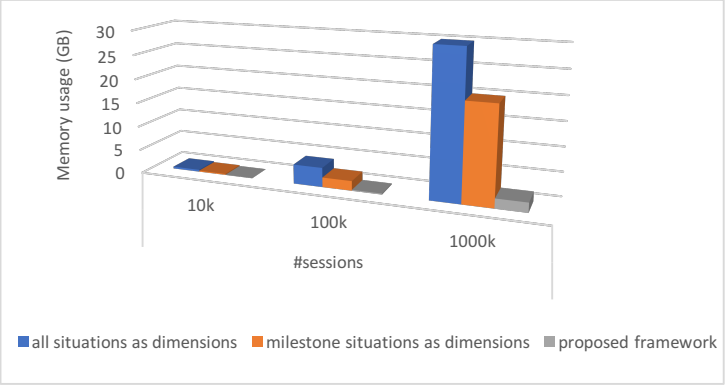



Figure 9.7 Memory usage for three different predictive analysis approaches

As shown in Figure 9.7, the proposed framework uses less space compare to two other approaches. The reason is that it reduces the dimensions to a low constant number so that the matrix of situations and sessions is smaller in size. In other words, for any two situation session vectors with m dimensions, their corresponding $MH_a$ and $MH_b$ vectors have k dimensions such that k << m with enough information to estimate similarity between sessions. For m be around 10,000, k can be any number from 400 to 1000. Therefore, instead of storing a vector with 10000 integers each taking 4 bytes, we can store the minhash signature with a 400-1000 integers. As a result, for a minHash signature with 1000 dimensions, it takes 4KB memory to be stored and 4GB to store one million sessions in memory. However, it is not feasible to store actual sessions with 10000 dimensions in memory.

**Query Speed**

In this part, we compare the time taken to find similar sessions to query session using exact similarity search and approximate similarity search used in this study. For example, to estimate similarity Type II the time taken to compute the MinHash matrix of a situation session collection is computed by adding the time of creating each column of matrix that is MinHash signature of a session. A minhash signature is computed by computing each of

$\min[\Pi_1(Ses_i)]$, …, $\min[\Pi_k(Ses_i)]$. We can compute $\min[\Pi_l(Ses_i)]$ by computing $\Pi_l$ on every situation in the session. Therefore, computation of $\min(\Pi_l(Ses_i))$ takes O(number of situations in $Ses_i$) time. Thus, the total time taken to compute the MinHash matrix is equal to $k \times$ (Number of situations in $Ses_1$+ Number of situations in $Ses_2$ +…+ Number of situations in $Ses_N$) and k*m is an upper bound on this value where m is the number of dimensions of original situation session vectors. After computing MinHash matrix, we can estimate similarity between any two sessions in linear time. The MinHash signature of each session is a k-tuple vector where k is number of permutations. And if $MH_a$ and $MH_b$ matches at L places then L/k is the estimation of similarity Type II. Therefore, we can estimate similarity of two sessions in O(k) because we just need to check k numbers if they match or not with another k numbers. The idea of LSH is to focus on pairs that are likely to be similar instead of focusing on every pair using locality sensitive hashing that maps “similar items" into the same bucket.

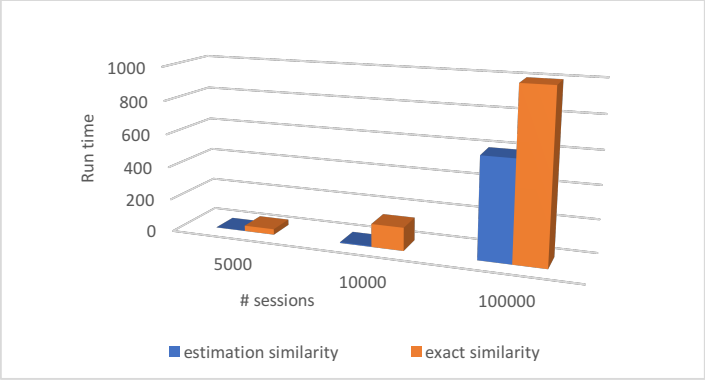



Figure 9.8 Run time similarity computation for all session pairs (in approach 1 and 2 exact similarity search is used, in proposed approach, the approximate similarity search is used) (given $\pi$= 600, dimensions = 30000)

As it is shown from Figure 9.8, time to compute similarity between sessions and find close sessions to a query session takes less time when it uses the approximate approach compared to the exact-pair similarity computation. The reason is that in naïve approach, we

compare the query session with all sessions to find the closes neighbors; however; using locality sensitive hashing, it searches among the pairs that are likely to be similar.

**Relevancy Result**

To evaluate the proposed framework in terms of relevancy of returned results to actual similar results, we measure precision with counting the number of pairs of sessions in which their difference is less than an error parameter. In other words, precision is percentage of correctly identified pairs as similar sessions over all pairs. We have tested our model with three different number of permutations and error parameters.

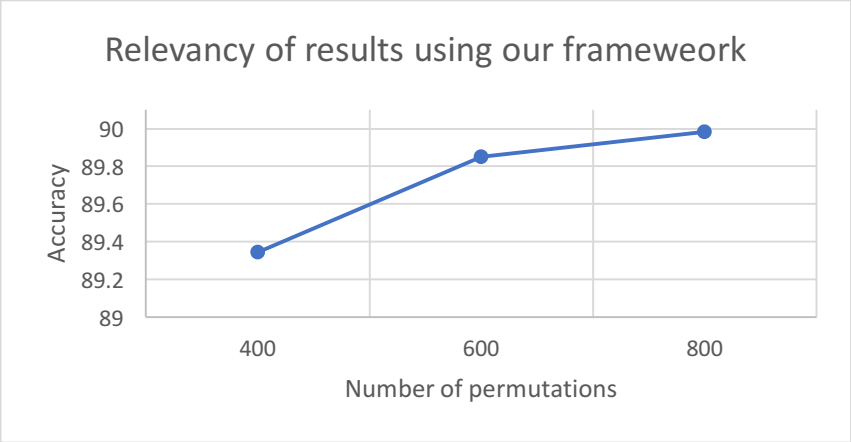



Figure 9.9 Accuracy of our framework given 100000 sessions, 30000 situations, total number of pairs: 4999950000 and error parameter 0.04

As Figure *9.9* shows, the accuracy of our proposed framework is around 89% by choosing appropriate parameters for locality sensitive hashing.

# CHAPTER 10. DISCUSSION AND CONCLUSION

In this thesis, we propose a data-driven situation-aware framework for predictive analysis in smart environments. The proposed framework has different components: scalable Situ ranking, Situ-Morphism, approximate nearest neighbor search technique. Situation data brings the addition of meta-information to raw sensor data that allows us to make more reliable decisions for smart environments. However, storing more data will improve the decision quality but cause more computation complexity. To address this issue, we propose a scalable iterative ranking method based on PageRank to assign a score to situations based on their importance and filter out the non-important situations from future predictive purposes. We apply MapReduce programming model to implement the proposed technique. The results of our experiments on both graphical predictive models and approximate predictive models show that using the scalable Situ ranking, we can make effective use of the situation data, leverage situation information on the situation graph, and scale up to very-large scale situation-aware predictive analysis problems. We evaluate the performance of the proposed ranking technique using two different predictive approaches: probabilistic graphical models (i.e. higher Markov models) and approximate nearest neighbor search techniques. In both approaches, the experimental results suggest the superior performance of predictive model combined with scalable Situ ranking over the other related predictive approaches. The reason of outperformance lies in the idea that it employs a ranking process to identify milestone situations (i.e. high-ranked) and transitions with respect to the user's current mental state in order to compress the transition matrix while keeping the accuracy high. In order to apply approximate nearest neighbor search techniques, we define a new concept called Situ-Morphism to map Situ space to vector space while preserving the structure of situation

sessions (i.e. milestone situations, similarity between sessions, composition of situations and their relative position). Using this morphism, we are able to utilize approximate nearest neighbor search techniques defined in vector space to find the nearest neighbors of a given session. Using locality sensitive hashing, the situation sessions are hashed down to a lower dimensional space so that they occupy less space and compare faster. Also, finding similar situation sessions to user's current session helps with predicting the future situations, extracting user's behavioral patterns, detecting anomalies and avoiding any hazard situation that may happen in future. We validate our framework on a dataset collected by simulating a smart home using OpenSHS tool. The results show that our framework improves the time and space for predictive analysis while keeping the relevancy of returned results high enough compared to naïve exact search approach or other existing predictive analysis frameworks. There are some limitations on this study that can be addressed in future work. In this study, we make some assumptions to keep the scope of the framework manageable. However, in real world applications, some of these assumptions may not apply. Depending on the users and the application, the data may be different and some noises apply. For example, if this framework is used for the users with dementia, there is more likely to observe situation sessions with no ending or sessions with many changes in desires. Also, sometimes identifying change in desire may become complicated because of the noise in data. The data used in this research is collected using a simulation tool considering all assumptions made. Another limitation of this work is the fact that we analyze user's situations with respect to desires and the first layer of similarity between two sessions is defined based on same desires. In real use cases, studying the patterns and discovering user's behavior with different desires may be valuable. For example, in sessions with respect to desire taking a shower and

sessions with respect to desire leaving home, the desire is different but the pattern for falling might be the same and it is valuable to be discovered. One other limitation is that the labelling of activities is performed by the participant during simulation phase. Barring the uses the simulation tool, automatic recognition of activities in real-life smart environments would improve the process of data collection in terms of time and accuracy. Another improvement would be to add more sensors to capture more situations and dimensions. High dimensionality has an unclear meaning sometimes. So, the higher the number of dimensions becomes, the better evaluation of model is expected. Another limitation of our work is the size of data. Generating big data in the context of smart environment is a challenge since it involves constantly interaction with the user and sensors. However, the data generated for this study are not considered big enough compared to data-driven real-world applications, suffice it to say that, should real-life big data sets for smart environments enabled by IoT be made accessible, we trust that our framework works well for bigger datasets.

## REFERENCES


[1] K. Ashton, "That 'internet of things' thing in the real world things matter more than ideas", RFID Journal June 2009, [online] Available: http://www.rfidjournal.comlarticle/orint/4986.

[2] A. Zaslavsky, C. Perera, D. Georgakopoulos, "Sensing as a service and big data", International Conference on Advances in Cloud Computing (ACC- 2012), pp. 21-29, July 2012.

[3] H. Sundmaeker, P. Guillemin, P. Friess, S. Woelffle, "Vision and challenges for realising the internet of things", Tech. Rep March 2010, [online] Available: http://www.internet-of-things-research.eu/pdf/IoT_Clusterbook_March_2010.pdf.

[4] D. A. Reed, D. B. Gannon, J. R. Larus, "Imagining the Future: Thoughts on Computing", IEEE Computer, vol. 45, no. 1, pp. 25-30, January 2012.

[5] O. Vermesan, P. Friess, P. Guillemin, "Internet of things strategic research roadmap", The Cluster of European Research Projects, 2009, [online] Available: http://www.internet-of-things research.eu/pdf/IoT_Cluster_Strategic_Research_Agenda_2009.pdf.

[6] K. benedict, "Moneyball, Big Data, The Internet of Things and Enterprise Mobility," 2012. [Online]. Available: https://sapinsider.wispubs.com/Assets/Blogs/2012/February/Moneyball-Big-Data-The-Internet-of-Things-and-Enterprise-Mobility. [Accessed: 29-Mar-2018].

[7] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos, "Context aware computing for the internet of things: A survey", IEEE Commun. Surv. Tuts., vol. 16, no. 1, pp. 414-454, Jan. 2014.

[8] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," IEEE Netw., vol. 8, no. 5, pp. 22–32, Sep. 1994.

[9] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, P. Steggles, "Towards a better understanding of context and context-awareness", Proc. 1st international symposium on Handheld and Ubiquitous Computing, pp. 304-307, 1999.

[10] H. Xie, C.K. Chang, "Detection of New Intentions from Users Using the CRF Method for Software Service Evolution in Context-Aware Environments", Proc. IEEE 39th Ann. Int'l Computers Software & Applications Conf., pp. 71-76, 2015.

[11] C. K. Chang, H. Jiang, H. Ming, and K. Oyama, “Situ: A Situation-Theoretic Approach to Context-Aware Service Evolution,” IEEE Trans. Serv. Comput., vol. 2, no. 3, pp. 261–275, Jul. 2009.

[12] H. Gholami, C. K. Chang, "Situation-Aware Decision Making in Smart Homes", International Conference on Smart Homes and Health Telematics, pp. 71-82, 2016.

[13] N. Alshammari, T. Alshammari, M. Sedky, J. Champion, and C. Bauer, “OpenSHS: Open Smart Home Simulator,” Sensors, vol. 17, no. 5, p. 1003, May 2017.

[14] H. Gholami and C. K. Chang, “Situation-Aware Data-Driven Decision Making in Smart Environments Using the MapReduce,” in 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), pp. 722–730, 2017.

[15] D. Wang, M. T. Amin, S. Li, T. Abdelzaher, L. Kaplan, S. Gu, C. Pan, H. Liu, C. C. Aggarwal, R. Ganti et al., "Using humans as sensors: an estimation-theoretic perspective", Proceedings of the 13th international symposium on Information processing in sensor networks, pp. 35-46, 2014.

[16] A. van Bunningen, L. Feng, P. Apers, "Context for ubiquitous data management", Ubiquitous Data Management 2005. UDM 2005. International Workshop on, pp. 17-24, april 2005.

[17] D. Evans, “The Internet of Things How the Next Evolution of the Internet Is Changing Everything,” Cisco White Paper, 2011.

[18] J. McCarthy, P.J Hayes, "Some Philosophic Problems from the Standpoint of Artificial Intelligence", Machine Intelligence, vol. 4, pp. 463-502, 1969.

[19] R. Reiter, “The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression,” in Artificial Intelligence and Mathematical Theory of Computation, pp. 359–380, 1991.

[20] S. S. Yau, Y. Wang, F. Karim, "Adaptable Situation-Aware Secure Service Based Systems", Proc. 8th IEEE Symp. on Object-oriented Real-time distributed Computing, pp. 308-315, 2005.

[21] K. Devlin, “Situation theory and situation semantics,” in Handbook of the History of Logic, vol. 7, pp. 601–664, Jan. 2006.

[22] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” Computer Networks, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[23] J. Augusto, C. Nugent, "Smart Homes Can Be Smarter," Lecture Notes in Computer Science, vol. 4008, pp. 1-15, 2006.

[24] M. Weiser and Mark, “The computer for the 21 st century,” ACM SIGMOBILE Mob. Comput. Commun. Rev., vol. 3, no. 3, pp. 3–11, Jul. 1999. M. Weiser, "The computer for the 21st century", ACM SIGMOBILE Mobile Computing and Communications Review, vol. 3, no. 3, pp. 3-11, 1999.

[25] K. Bouchard, S. Gaboury, B. Bouchard, A. Bouzouane, S. Giroux, K.K. Ravulakollu, A.M. Khan, A. Abraham, "Smart Homes in the Era of Big Data" in Trends in Ambient Intelligent Systems: The Role of Computational Intelligence, Springer International Publishing, pp. 117-137, 2016.

[26] N. K. Suryadevara, S. C. Mukhopadhyay, R. Wang, and R. K. Rayudu, “Forecasting the behavior of an elderly using wireless sensors data in a smart home,” EngineeringApplications of Artificial Intelligence, vol. 26, no. 10, pp. 2641–2652, Nov. 2013.

[27] P. Boissy, S. Choquette, M. Hamel, and N. Noury, “User-Based Motion Sensing and Fuzzy Logic for Automated Fall Detection in Older Adults,” Telemed. e-Health, vol. 13, no. 6, pp. 683–694, Dec. 2007.

[28] X. Yu, "Approaches and principles of fall detection for elderly andpatient", Proceeding of 10th International Conference on e-health Networking, Applications and Services, HealthCom, pp. 42-47, 2008.

[29] MJ Akhlaghinia, A Lotti, C Langensiepen, N Sherkat, "A fuzzy predictor model for the occupancy prediction of an intelligent inhabited environment", 2008 IEEE 16th International Conference on Fuzzy Systems (FUZZ-IEEE) London, pp. 939-946, 2008.

[30] V. Callaghan, G. Clarke, M. Colley, H. Hagras, J. Chin, and F. Doctor, "Inhabited intelligent environments," BT Technology Journal, Vol. 22, no. 3, pp. 233-247, 2004.

[31] H. Hagras, “Embedding Computational Intelligence in Pervasive Spaces,” IEEE Pervasive Computuing, vol. 6, no. 3, pp. 85–89, Jul. 2007.

[32] A. S. Helal, J. King, R. Bose, E.-Z. Hicham, and Y. Kaddourah, "Assistive environments for successful aging, " in Advanced Intelligent Environments, ed: Springer, 2009, pp. 1-26.

[33] A. Serna, H. Pigot, and V. Rialle, “Modeling the progression of Alzheimer’s disease for cognitive assistance in smart homes,” user modeling and user-adapted interaction, vol. 17, no. 4, pp. 415–438, Aug. 2007.

[34] N. Noury and T. Hadidi, “Computer simulation of the activity of the elderly person living independently in a Health Smart Home,” Computer Methods and Programs in Biomedicine, vol. 108, no. 3, pp. 1216–1228, Dec. 2012.

[35] H. Li, Q. Zhang, P. Duan, "A Novel One-Pass Neural Network Approach for Activities Recognition in Intelligent Environments", Proceedings of the 7th World Congress on Intelligent Control and Automation, pp. 50-54, 2008.

[36] H. Zheng, H. Wang, N. Black, "Human activity detection in smart home environment with self-adaptive neural networks", Proc. IEEE Int. Conf. Netw. Sensing Control, pp. 1505-1510, 2008.

[37] F. Rivera-Illingworth, V. Callaghan, H. Hagras, "Towards the detection of temporal behavioural patterns in intelligent environments", 2nd IET International Conference Intelligent Environments (IE 06), vol. 2006, no. July, pp. v1-119-v1-119, 2006.

[38] F. Rivera-Illingworth, V. Callaghan, H. A. Hagras, "A neural network agent based approach to activity detection in AmI environments", IEE Seminar on Intelligent Building Environments IEE Seminar Digests, vol. v2-9, 2005.

[39] E. Tapia, S. Intille, K. Larson, "Activity recognition in the home using simple and ubiquitous sensors", Proc. 2nd Int. Conf. Pervasive Comput., pp. 158-175, 2004.

[40] L. Chen, C. D. Nugent, and H. Wang, “A Knowledge-Driven Approach to Activity Recognition in Smart Homes,” IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 6, pp. 961–974, Jun. 2012.

[41] O. Kwon, J. M. Shim, and G. Lim, “Single activity sensor-based ensemble analysis for health monitoring of solitary elderly people,” Expert Systems with Applications, vol. 39, no. 5, pp. 5774–5783, Apr. 2012.

[42] Cook DJ, V. Jakkula, Crandall AS, "Temporal pattern discovery for anomaly detection in a smart home", Proceedings of the 3rd IET Conf. on Intelligent Environments(IE), pp. 339-345, 2007.

[43] D. N. Monekosso, N. Dorothy, P. Remagnino, "Anomalous Behavior Detection: Supporting Independent Living", Intelligent Environments, pp. 33-48, 2008.

[44] H. Kautz, O. Etziono, D. Fox, D. Weld, "Foundations of assisted cognition systems", technical report cse-02-ac-01 University of Washington Department of Computer Science and Engineering, 2003.

[45] O. Brdiczka, J. L. Crowley, and P. Reignier, “Learning Situation Models in a Smart Home,” IEEE Transactions on Systems, Man, and Cybernetics, Part B, vol. 39, no. 1, pp. 56–63, Feb. 2009.

[46] D. Sanchez, M. Tentori, and J. Favela, “Activity Recognition for the Smart Hospital,” IEEE Intelligent Systems, vol. 23, no. 2, pp. 50–57, Mar. 2008.

[47] L. Bao, S. Intille, "Activity recognition from user-annotated acceleration data", Proc. 2nd Int. Conf. Pervasive Computing, pp. 1-17, 2004.

[48] P. Raghavan, R. Motwani, "Randomized Algorithms," Cambridge, U.K.:Cambridge Univ. Press, 1995.

[49] M. Eirinaki and M. Vazirgiannis, “Usage-based PageRank for Web personalization,” Proc. - IEEE International Conference on Data Mining, ICDM, pp. 130–137, 2005.

[50] S. Brin, L. Page, S. Brin, and L. Page, “The anatomy of a large-scale hypertextual Web search engine,” Computer Networks ISDN Systems, vol. 30, no. 1–7, pp. 107–117, Apr. 1998.

[51] M. Aktas, M. Nacar, F. Menczer, "Personalizing PageRank based on Domain Profiles", in Proceedings of the 6 WEBKDD Workshop, 2004.

[52] T.H. Haveliwala, "Topic-Sensitive PageRank", Proc. 11th International World Wide Web Conference, May 2002.

[53] M. Richardson and P. Domingos, "The intelligent surfer: probabilistic combination of link and content information in PageRank," In Advances in Neural Information Processing Systems 14, pp. 1441-1448. MIT Press, 2001.

[54] I. Satoh, "A Framework for Data Processing at the Edges of Networks,"Database and Expert Systems Applications, vol. 8056, pp. 304-318, 2013.

[55] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, B. Moon, "Parallel data processing with mapreduce: A survey", Association for Computing Machinery (ACM) SIGMOD Record, vol. 40, no. 4, pp. 11-20, 2012.

[56] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Proc. of the 6th Symposium on Operating Systems Design and Implementation San Francisco CA, Dec. 2004.

[57] J. Lin, C. Dyer, "Data-intensive text processing with mapreduce", Synthesis Lectures on Human Language Technologies, vol. 3, no. 1, pp. 1-177, 2010.

[58] C.T. Chu, S.K. Kim, Y.A. Lin, Y. Yu, G.R. Bradski, A.Y. Ng, K. Olukotun, "Map-Reduce for Machine Learning on Multicore," in Proceeding 20th Annual Conferenece Neural Information Processing Systems (NIPS '06), pp. 281-288, 2006.

[59] B. Panda, J. Herbach, S. Basu, R. J. Bayardo, "Planet: Massively parallel learning of tree ensembles with mapreduce", PVLDB, vol. 2, no. 2, pp. 1426-1437, 2009.

[60] J. Wolfe, A. Haghighi, and D. Klein, “Fully distributed EM for very large datasets,” in Proceedings of the 25th international conference on Machine learning - ICML ’08, pp. 1184–1191, 2008.

[61] T. Elsayed, J. Lin, Douglas W. Oard, "Pairwise Document Similarity in Large Collections with MapReduce", Proceedings-32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR, 2009.

[62] T. Brants, A.C. Popat, P. Xu, F.J. Och, and J. Dean, "Large language models in Machine Translation," in EMNLP, 2007.

[63] . Dyer, A. Cordova, A. Mont, and J. Lin, "Fast, easy, and cheap: construction of statistical machine translation models with mapreduce," in Proceedings of the Third Workshop on Statistical Machine Translation, ser. Stat MT '08. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 199-207, 2008.

[64] J. Lin and Jimmy, “Brute force and indexed approaches to pairwise document similarity comparisons with MapReduce,” in Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval - SIGIR ’09, p. 155, 2009.

[65] B. Langmead, M. C. Schatz, J. Lin, M. Pop, and S. L. Salzberg, “Searching for SNPs with cloud computing,” Genome Biology, vol. 10, no. 11, p. R134, 2009.

[66] M. C. Schatz, “CloudBurst: highly sensitive read mapping with MapReduce,” Bioinformatics, vol. 25, no. 11, pp. 1363–1369, Jun. 2009.

[67] B. Gao, T.-Y. Liu, W. Wei, T. Wang, and H. Li, “Semi-supervised ranking on very large graphs with rich metadata,” in Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’11, p. 96, 2011.

[68] D. Rao, D. Yarowsky, "Ranking and Semi-supervised classification on large scale graphs using map-reduce", Proc. Workshop Graph-Based Methods Natural Language Process., pp. 58-65, 2009.

[69] I. Satoh, “MapReduce-Based Data Processing on IoT,” in 2014 IEEE International Conference on Internet of Things(iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom), pp. 161–168, 2014.

[70] I. Miller, “Probability, Random Variables, and Stochastic Processes,” Technometrics, vol. 8, no. 2, pp. 378–380, May 1966.

[71] M. R. Alam, M. B. I. Reaz, and M. A. Mohd Ali, “SPEED: An Inhabitant Activity Prediction Algorithm for Smart Homes,” IEEE Transactions on Systems, Man, and Cybernetics -Part A, vol. 42, no. 4, pp. 985–990, Jul. 2012.

[72] P. Chahuara, F. Portet, M. Vacher, "Making Context Aware Decision from Uncertain Information in a Smart Home: A Markov Logic Network Approach" in Ambient Intelligence ser. Lecture Notes in Computer Science, Dublin, Ireland: Springer, vol. 8309, pp. 78-93, 2013.

[73] P. Chahuara, A. Fleury, F. Portet, and M. Vacher, "Using Markov Logic Network for On-line Activity Recognition from Non-Visual Home Automation Sensors, " in Ambient Intelligence, Pisa, Italy, pp. 177-192, 2012.

[74] M. Deshpande, G. Karypis, "Selective Markov Models for Predicting Web-Page Accesses", Proc. SIAM Int'l Conf. Data Mining (SDM '01), Apr. 2001.

[75] J. Pitkow and P. Pirolli, “Mining longest repeating subsequences to predict world wide web surfing,” Proceedings of the 2nd conference on USENIX Symposium on Internet Technologies and Systems - Volume 2. USENIX Association, pp. 13–13, 1999.

[76] M. Eirinaki, M. Vazirgiannis, D. Kapogiannis, "Web Path Recommendations Based on Page Ranking and Markov Models", Proc. Seventh Ann. ACM Int'l Workshop Web Information and Data Management (WIDM '05), pp. 2-9, 2005.

[77] J. Zhu, J. Hong, and J. G. Hughes, "Using Markov chains for link prediction in adaptive web sites", Proceeding of soft-ware: first international conference on computing in an imperfect world, Belfast, UK, pp. 60-73, 2002.

[78] M. H. Rituparna Sen, “Predicting Web User’s Next Access Based on Log Data,” Journal of Computational and Graphical Statistics , pp. 143–155, 2003.

[79] J. Borges, M. Levene, "Data mining of user navigation patterns", Proceedings of the Workshop on Web Usage Analysis and User Profiling, San Diego CA USA, Aug 1999.

[80] J. Wang, Z. Chen, L. Tao, W. Ma, and W. Liu, "Ranking user's relevance to a topic through link analysis on web logs," WIDM, pp. 49-54, 2002.

[81] R. R. Sarukkai, “Link prediction and path analysis using Markov chains,” Comput. Networks, vol. 33, no. 1–6, pp. 377–386, Jun. 2000.

[82] I. V. Cadez, S. Gaffney, and P. Smyth, “A general probabilistic framework for clustering individuals and objects,” in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’00, pp. 140–149, 2000.

[83] J. Leskovec Stanford Univ Anand Rajaraman, J. D. Ullman, A. Rajaraman, J. Leskovec, and J. D. Ullman ii, “Mining of Massive Datasets,” 2010.

[84] Y. Li, Z. Bandar, D. McLean, J. O'Shea, "A Method for Measuring Sentence Similarity and its Application to Conversational Agents", Proceeding17th Int. Florida AI Research Society Conf. (FLAIRS 2004), pp. 820-825, 2004.

[85] P. Indyk, R. Motwani, "Approximate nearest neighbor: Toward removing the curse of dimensionality", Proc. 30th Annuual ACM Symposium on Computational Geometry, 1998.

[86] J. Wang, H. T. Shen, J. Song, J. Ji, "Hashing for similarity search: A survey", CoRR, 2014.

[87] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, “Locality-sensitive hashing scheme based on p-stable distributions,” in Proceedings of the twentieth annual symposium on Computational geometry - SCG ’04, p. 253, 2004.

[88] M. Charikar, "Similarity Estimation Techniques from Rounding Algorithms", ACM Symposium on Theory of Computing, 2002.

[89] A. Broder, "On the Resemblance and Containment of Documents", Proc. Compression and Complexity of Sequences Conf. (SEQUENCES '97), pp. 2129, 1998.

[90] A. Z. Broder, S. C. Classman, M. S. Manasse, G Zweig, "Syntactic clustering of the web", Proc. of 6th International World Wide Web Conference, 1997.

[91] A. Dasgupta, R. Kumar, and T. Sarlos, “Fast locality-sensitive hashing,” in Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’11, p. 1073, 2011.

[92] R. Motwani, A. Naor, and R. Panigrahy. Lower bounds on locality sensitive hashing. Proceedings of the ACM Symposium on Computational Geometry, 2006.

[93] R. O'Donnell, Y. Wu, Y. Zhou, "Optimal lower bounds for locality sensitive hashing (except when q is tiny)", Proc. Int. Conf. Supercomputing, pp. 275-283, 2011.

[94] F. Chierichetti and R. Kumar, “LSH-preserving functions and their applications,” Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, pp. 1078–1094, 2012.

[95] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In SODA, pages 1186-1195. ACM, 2006.

[96] E.Viehweg, "Rational singularities of higher dimensional schemes," in Proceeding AMS.63 no.1, pp.6-8, 1977.

[97] A. Shrivastava, P. Li, "In Defense of Minhash Over SimHash", AISTATS, pp. 886-894, 2014.

[98] A. Z. Broder, M. Charikar, A. M. Frieze, M. Mitzenmacher, "Min-Wise Independent Permutations", Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, 1998.

[99] D. J. Cook, "Learning setting-generalized activity models for smart spaces", IEEE Intelligent Systems, vol. 27, no. 1, pp. 32-38, Jan./Feb. 2012.

[100] E. Nazerfard and D. J. Cook, “CRAFFT: an activity prediction model based on Bayesian networks,” Journal of Ambient Intelligence and Humanized Computing, vol. 6, no. 2, pp. 193–205, Apr. 2015.

[101] C. Dobre and F. Xhafa, “Parallel Programming Paradigms and Frameworks in Big Data Era,” International Journal parallel programming, vol. 42, no. 5, pp. 710–738, Oct. 2014.