*DRAFT*

Master thesis report for the MSc Embedded Systems

TU Delft – Interactive Intelligence

Notival

User valued smart reminders: Finding Appropriate Moments for Support in Socially Adaptive Electronic Partners

Remy Kabel

4132165

Supervised by:  
Dr. M. Birna van Riemsdijk  
Dr. Myrthe Tielman

09-01-2019

# Abstract

This project will focus on finding what defines an appropriate moment in regards to providing support through a Socially Adaptive Electronic Partner (SAEP). We focus specifically on designing a dynamic, smart notification system. To determine the most ideal moment for a notification, in this project we investigate how knowledge about user values can be used for this. The idea is then that ideality is determined through the effects of interruption on user values, as defined by the (expected) gains (from remembering to do the task) and losses (from being interrupted) in user values. To achieve this, we combine existing knowledge of activity prediction models and value based design. In summary, the main research question of the thesis is as such: "How can existing smart reminder systems be extended to incorporate user values to provide appropriately timed supportive feedback and thereby increase the user values.”

**Rewrite**

# Table of common terms

|  |  |
| --- | --- |
| **Term** | **Description** |
| ADL | Activities of daily living |
|  |  |
| SAEP | Socially Adaptive Electronic Partner |
| Middleware | Software layer that acts as acts a link between two layers by processing data before it is passed from one to the other. |
|  |  |
| Markov chain | Probabilistic model describing a sequence of events based solely on the state attained in the previous event. |
|  |  |
| Clustering | A method of grouping data points according to an algorithm |
|  |  |
| Route | And endpoint (or address) for an HTTP request |
|  |  |
| API | Application Programming Interface. A set of definitions used among applications to communicate between one another. |
|  |  |
| RESTful | An API standard based on representational state technology (REST). A standardized, architectural approach web communication using HTTP methodologies: GET, POST, PUT, DELETE |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Table of Contents

[Abstract 2](#_Toc535005328)

[Table of common terms 3](#_Toc535005329)

[Table of Contents 4](#_Toc535005330)

[1 Introduction 5](#_Toc535005331)

[1.1 Problem description 6](#_Toc535005332)

[2 Related Work 7](#_Toc535005333)

[2.1 Existing implementations 7](#_Toc535005334)

[2.2 Literature study 7](#_Toc535005335)

[3 Approach 9](#_Toc535005336)

[3.1 Starting point 9](#_Toc535005337)

[3.2 Research questions 10](#_Toc535005338)

[3.3 Roadmap 11](#_Toc535005339)

[4 Requirement analysis 12](#_Toc535005340)

[4.1 Model requirements 12](#_Toc535005341)

[4.2 Existing implementations 13](#_Toc535005342)

[4.3 Comparison 17](#_Toc535005343)

[4.4 User values 19](#_Toc535005344)

[4.4.1 Which values 19](#_Toc535005345)

[4.5 Data acquisition 19](#_Toc535005346)

[4.6 Conclusions 21](#_Toc535005347)

[5 Concept design 22](#_Toc535005348)

[5.1 High-level overview 22](#_Toc535005349)

[5.2 Processing incoming data 23](#_Toc535005350)

[5.2.1 Dataset 23](#_Toc535005351)

[5.2.2 Data stream 24](#_Toc535005352)

[5.3 Activity prediction 24](#_Toc535005353)

[5.3.1 Expectation Maximization 24](#_Toc535005354)

[5.3.2 Apriori algorithm 25](#_Toc535005355)

[5.4 Value based design 27](#_Toc535005356)

[5.4.1 The appropriate time 27](#_Toc535005357)

[5.4.2 How to measure? 28](#_Toc535005358)

[5.5 Statistical analysis 28](#_Toc535005359)

[5.5.1 Expected value 29](#_Toc535005360)

[5.5.2 Absorbing Markov chain 31](#_Toc535005361)

[5.5.3 Drawback of choosing Markov chains 34](#_Toc535005362)

[5.6 Concept description 35](#_Toc535005363)

[6 Implementation 36](#_Toc535005364)

[6.1 Platform 36](#_Toc535005365)

[6.1.1 Internet of Things 36](#_Toc535005366)

[6.1.2 Programming language 37](#_Toc535005367)

[6.1.3 Set-up 37](#_Toc535005368)

[7 System architecture 38](#_Toc535005369)

[7.1 Schematics 38](#_Toc535005370)

[7.2 API 38](#_Toc535005371)

[8 Experimentation 39](#_Toc535005372)

[8.1 Testing 39](#_Toc535005373)

[8.2 Results 39](#_Toc535005374)

[9 Discussion 40](#_Toc535005375)

[10 References 41](#_Toc535005376)

[11 Appendices 46](#_Toc535005377)

[11.1 Unique activities in dataset 46](#_Toc535005378)

# Introduction

The use of technology to support the daily lives of people is an ever-prevalent topic. Through applications in smart homes, wearables, virtual coaches and many others, we can improve our health, efficiency and be more connected. Conversely, the abundance of apps and notifications causes us to grow immune to the constant stream of information that is presented to us in a daily basis [1]. Especially the elderly or people with a mental impairment could benefit from an effective support agent [2]–[7]. In order to create a truly effective support agent, it is crucial to not only generate feedback in relation to the user’s actions but to provide this feedback at an appropriate time.

But what actually is an appropriate time? The appropriate time for feedback is inherently linked to the nature of the user’s action. To illustrate this, consider the following example throughout this report.

An elderly gentleman, Peter, often forgets to close the garden doors before leaving the house or going to sleep.

**Visual/Plaatje!**

In this example, timely notification is of the essence. Preferably, notification just before sleeping or leaving the house is desired. Generally, these are quite predictable activities. In the current technological landscape, a simple scheduled notification would be the likely solution. Possibly a geofence[[1]](#footnote-1) may be used to trigger a notification when leaving the house, but this will be post factum.

Identifying such an appropriate time for support feedback for a specific scenario is not difficult. The difficulty of this lies in the generalization. While the above examples can be implemented relatively easy at design time, diversions from normal behavior are not handled. Existing technologies are often made by hardwiring norms and as such are very rigid and unable to adapt to evolving norms [8]. Furthermore, dealing with different problems, such as remembering to turn on the alarm system before leaving work, would require a completely different implementation. Nonetheless, generalization requires analysis of goals and the values underlying the user’s daily activities.

## Problem description

The problem of finding boils down to a few steps; each worth further analysis in their own right. Working our way back, the first question that arises is what defines the goal. The goal is defined by the users and can be anything such as: *“I want my garden doors to be closed when I go to sleep or leave the house”.* Assuming we know the user’s activities of daily living (ADL), and optionally the status of the garden doors at any moment, the first step is analyzing which prerequisites there are to attaining that goal. Usually, a goal is not an independent action taken, but rather the consequence of a series of actions. As such, knowledge is required on how a goal be deconstructed into a number of distinct prerequisites.

In order to analyze arriving at this goal, some sort of model needs to be created from the user’s ADL. Once this model has been created, we can use it to analyze the limits of the possible moments for support. More directly, the prerequisites will indicate a number of actions that will have to have been completed, but also some actions may not have been completed. For example, a user will first have to arrive home, but should have received the support feedback before leaving once again, when the user will need their keys. However, Finding the most suitable moment for support is dependent on more than just this.

Finding the most “appropriate” time for the support feedback boils down to finding a moment which is both maximally effective and minimally invasive. Depending on the chosen solution, a number of other values are negatively affected. For example, sounding an alarm in the middle of a person’s sleep may be very effective, but it sure is annoying. The problem is, however, that it’s difficult to quantify invasiveness.

Summarizing, the required steps are:

* Definition of the goal and its prerequisites
* Analysis and modelling of the user’s ADL
* Analysis of effectivity
* Analysis of invasiveness

**Herschrijven naar wat we daadwerkelijk doen in dit project**

*(This should, however, be analyzed with respect to the consequences of not remembering.) In case Peter forgets before sleeping, he will either wake up with a sense of insecurity, or if he wakes up at night, he will have to get out of bed and properly interrupt his sleep. If he forgets and leaves the house, the only solutions may be to return home, to ask a friend, or to leave it be. In all cases, his value of security will be diminished, let alone if a break-in were to actually happen.* ***Dit moet nog ergens, toch?***

# Related Work

There are plentiful existing implementations, related papers and interesting concepts. This chapter revolves around those existing and past works, in service of finding an approach to the aforementioned problems.

## Existing implementations

More and more apps are taking advantage of the increased use of smart devices and services in order to get a more accurate picture of the user’s ADL. The following examples are finished

Olisto/IFTTT [9], [10] Can combine date, location and smart device information to, for example, give reminders when leaving home and a specific power consumption is still high (i.e. the TV is still on) and subsequently turn it off.

Maps/Waze [11]–[13] Combines real-time traffic information and address in calendar events to provide timely departure reminders.

Timeful [14] Combines user activity, calendar and to-do items to estimate duration of to-do items, plan them in and generate reminders at off-peak times.

While very promising implementations, most apps predominantly rely on design time logic. Exceptions to this usually create a predictive model and verify this with the user in order to strengthen the model [14], [15].

## Literature study

There have been various approaches as to how and when to provide feedback to the user. Generally, the preferred method of feedback is “smart reminders” [16]. Similar to the implementations, papers frequently focus on finding novel ways of combining information from smart devices into producing reminders, following norms provided at design time. Examples include combinations of location and time [17]–[19], events based on smart devices [3], [20], [21], or a combination of numerous sources of information [22]–[24].

The more innovative ideas add an extra logic layer on top of the data of the user’s ADL. Analyzing the user’s values is an intrinsic part of establishing a model. The concept of a Socially Adaptive Electronic Partner (SAEP) has been previously introduced by van Riemsdijk [8]. It follows the ideology that technology should adapt to the user and not vice versa. As such, its logic incorporates the norms and values of the social context. Subsequent work has been done expanding on this, including temporal logic and analyzing actions and habits. [25]–[27]. A simple but tedious approach is to ask for user feedback whenever values are needed. Instead, Zhou et al. [28] use a fuzzy linguistic approach to determine value levels.

Rather than specifying norms at design time, they are constructed based on the ADL. Several approaches are proposed. Chaminda et al. [29] suggest coupling complex activities that have a strong relationship among initiation and conclusion, such as closing the tap after opening it. Other papers [2], [30] support this analysis of temporal relationships between activities, in order to generate a set of norms for the support agent. Other context-aware approaches vary greatly. For example, Vurgun et al. [31] apply a dynamic Bayesian statistical approach. Giorgini et al. [32] use label propagation algorithms to break down goals and identify all prior actions necessary to achieve the goal.

Another approach for this makes use of Behavior Change Support Systems (BCSS) [33] by applying principles of Human Computer Interaction (HCI) [34]. This practice is used increasingly in health focused applications to make sense of the abundance of data. Examples of applications [35], [36] share large similarities with the analysis of the user’s norms and values.

# Approach

As previously mentioned, there are several steps in finding an appropriate time for supportive feedback. However, time is limited and several aspects have already been researched plenty. As such, let us limit the focus of the thesis research.

The first two steps, goal definition and ADL analysis are all linked to activity recognition and analysis to provide smart reminders. As discussed in the previous paragraph, several models covering exactly this already exist. Each having their own properties, advantages and disadvantages.

The effectiveness and invasiveness are both quite difficult to quantify. However, they can be combined into user values. These, in return, are more quantifiable. To illustrate this, let us revisit the example of the elderly man, Peter. The goal, closing the garden doors in time, clearly promotes his value of safety. The moment the supportive feedback is provided, however, may demote that value or another. For example, if it causes him to wake up from his sleep, it will demote his value of health, or if it interrupts him during a phone call it may demote his value of social contact.

## Starting point

Prior to being able to establish the research questions, the starting point needs to be established. This is also to limit the scope of the research since the general topic is very broad.

The area of activity recognition is a rapidly evolving one. However, the current state is that any forms of activity recognition based on raw sensor data are still very limited or inaccurate in general solutions. Accuracy can be improved by having location specific setups, or a severely limited number of recognized activities. Over the coming years, quality and accuracy of activity recognition is expected to increase thanks to, among others, the exponential rise in IoT devices in houses and public building [37] providing more and different data, as well as the improved sensors in and capabilities of smartphones.

Even partly focusing on actual activity recognition would therefore be an enormous enlargement of the scope of this thesis. As such, a choice is made to either use existing datasets which already contain information about a user’s ADL, or to make use of existing implementations that have proven to correctly provide streaming data about a user’s ADL. However, no focus should be put on actually analyzing the sensor data.

## Research questions

Combining the previous matters and assumptions, the focus of this thesis will be combining the concepts of a SAEP and expanding on the existing research as discussed before. The overall research question is:

How can existing smart reminder systems be extended to incorporate user values to provide appropriately timed supportive feedback and thereby increase the user values.

The expected outcome of this question is a model which provides timed feedback based on the user’s ADL and value input. Subsequently this leads to a number of sub-questions that need to be answered before this:

R1: What are the requirements for the smart reminder system model?

R2: Which existing models and systems exist for smart reminder systems and how do they compare.

These two questions should provide a good overview on the abilities of the existing systems and the amount of work required to extend them to incorporate user values. Of course, for this we need to be able to actually find out about the user values.

R3: What are possible ways of analyzing and quantifying the values of the user?

R4: How can the model be extended to incorporate user values?

Ultimately, all knowledge can be combined into a model which can be used to approximate the most “appropriate time” for support feedback. This model can subsequently be implemented in a piece of software in order for the model to be dynamically generated depending on new input regarding the ADL, goals, norms and values. Once such an implementation has been made, the model can be tweaked according to findings and should be tested. This brings us to the final sub-question:

R5: Does the use of the extended model improve support for user values?

This will require prior planning of possible testing methods and clearly defined testing scenarios.

## Roadmap

#### Requirement analysis

An extension of the preliminary research, the research questions are translated into requirements. This focuses on analyzing papers and implementations related to answering the research questions. Specifically, this revolves around analyzing and comparing past papers and reports to see possible ways of doing activity prediction, analyzing user goals and values, and ultimately combining them. All concepts should be compared on these previously established requirements in order to quickly identify the most valuable papers.

#### Concept design, architecture & implementation

Based on the findings of the requirement analysis, an initial concept can be designed. Accordingly, all required sub-components are analyzed and possible options for their implementations are discussed, leading to a final, defined design for the system architecture.

#### Experimentation and evaluation

Once the complete methodology has been established, the implementation can be tested upon actual data. This requires three things: a completed implementation, a suitable dataset and a method of evaluation.

# Requirement analysis

Within this chapter, we aim to answer the first four research sub-questions and arrive at a basic idea of how to design an initial concept. **In final design doe ergens wolkje met de RQ’s** First, all existing concepts will be compared according to a number of ideal requirements. Furthermore, the feasibility of all papers posing these concepts will be analyzed. Consecutively, analysis is done to the purpose of user values within this field, These are necessary before a model can be created in which the answers to these questions can be combined with the fourth into a concept design aimed at answering the fifth sub-question.

## Model requirements

Ideally, a model should notify at precisely the right time. Depending on whether the concept will be used as a middleware or as an full solution for the user, it should either intercept the existing notifications and only forward them to the user when desired, or only produce the notification when it is desired in the first place. This depends on whether the model incorporates goal reasoning or whether the reminder still has to be hardcoded.

**Give an example of why…**

Secondly, the model should incorporate user and environment values as much as possible. Initially utilization of activity information should be sufficient (because of its more direct link to user values), but preferably all user and environment variables should be considered.

If the goal would be “remind me to take my umbrella when I leave”, it’s nice to receive a notification when leaving the house, but preferably only when it’s actually going to rain.

Furthermore, as established earlier, we desire a more generalized solution rather than one mostly conceived at design time. As such, the solution should be dynamic through, for example, machine learning.

**A model that…**

Lastly, the model should preferably already have a notion of user values or at least be easily extendable to incorporate them.

**Does this need an example?**

## Existing implementations

In this section, the concepts previously mentioned in sections 2.2 and 2.3, as well as several others, are analyzed and compared to the aforementioned requirements. The papers mentioned all focus on one or more of the following aspects: activity prediction, smart reminders, goal reasoning or user values. Ultimately, an implementation is desired which combines all four of those aspects, or at least several of them.

**Deze het liefste allemaal in kolommetjes**

#### AHCS/TAFETA [24], [4]:

These concepts attempt to design a context-aware application which analyses data from various sensors within the user’s house. AHCS makes use of the CASanDRA framework [38] in order to create awareness of the user’s context. The CASanDRA framework is a middleware which provides easily consumable context information and accepts different information inputs which are fused together. The concepts use either the middleware or their own AI to analyze the collected information and compare this with a number of predefined rules to provide detailed information on the user to the caregiver and provide reminders when rules are broken.

Special properties:

* Context analysis independent from reminder system
* Levels and types of alerting

#### CogKnow [22]:

This concept actually touches upon user values, but instead uses them to define the required support. A distinct number of support scenarios are handled and rulesets are defined accordingly. Predominantly the user context is considered rather than anything else. The rulesets are aimed at avoiding interruptions of important activities, but don’t do any further analysis.

#### Gate reminder [20]:

This concept centralizes around providing reminders at the moment a user leaves their house. Knowledge about possibly forgotten items is obtained through the use of RFID tags, focusing on a zero user workload interaction. A crucial part in its working is that it is focused on Korean household, where shoes are generally left at the front door, so there is a clearly defined time slot in which all tags can be analyzed. Focus on the study was mostly the actual prototype rather than any smart algorithm.

*Special properties:*

* Physical prototype
* Transparent interaction
* Object detection

#### Goal models [32]:

This concept does not directly involve itself with reminders, but rather with linking certain activities to achieving certain goals. These activities may have complex relations with one another and may promote or demote a goal. As such, this can be similarly applied to activities aiming to achieve a certain goal where the promotions and demotions are linked to the user values.

Special properties:

* Linking activities to goals
* Not related to reminders

#### HeadacheCoach [35]:

While not directly a reminder system, HeadacheCoach does propose a possibly usable system. It uses user and environmental context analysis to identify possible triggers for a headache and consequently provides possible solution. A similar approach may be used to identify moments of lower cognitive ability in order to preempt a reminder being necessary at all.

#### MagHive [39]:

This honeycomb shaped magnetic smart surface is attached to the wall and allows devices and other objects to be placed on them. Aside from the useful functionalities such as wireless phone charging, it uses NFC and QI technologies to detect the presence and identity of the objects. As such it is able to remind the user when he or she forgets to take or put back an item.

Special properties:

* Actual product
* Provides a great base for further development

#### MLCARS [40]:

This dissertation discusses a concept which uses machine learning to analyze shopping items and where they were bought (or cleared off the to-do list) to predict similar available items or similar stores. This data is collected among all users and combined with information from companies and stores and ultimately stored in a database which is continuously updated. Combining this with the data of the user’s shopping list as well as their location allows to provide appropriately timed reminders not to forget items from their shopping list. These reminders are not just when near their usual supermarket (like is already possible with location-based reminders) but also when close to any store that is expected to have the desired item.

Special properties:

* Activity clustering
* Prediction of next activity without machine learning

#### Olisto/IFTTT/CAMP/CybreMinder [9], [10], [31], [41]:

These apps and concepts allow setting reminders based on various aspects of user and environment contexts. Once the current situation satisfies all conditions in all contexts, the user is automatically notified. Information is retrieved from the user’s (IoT) devices and (online) services. No form of pattern recognition or prediction is done, however.

Special properties:

* Existing (possibly discontinued) apps

#### Smart reminder system [29]:

This concept creates a smart reminder system through three major components: activity recognition, location recognition and prediction. The activity recognition is done through the use of analysis of the hand movements over time and applying machine learning algorithms and fuzzy logic to map this to activities. Location recognition is done through image recognition by camera and neural networks. These two are then combined to analyze coupled activities, two activities that are strongly related. Alongside, predictions are made regarding pending and forgotten activities. As such reminders can be produced when likely to be forgotten activities should occur.

Special properties:

* Specific setup

#### Attelia [1]:

Attelia is a middleware concept which intercepts any notifications. It analyses breakpoints in the user’s mobile interactions and adaptively delivers the notification to minimize interruptions and the user’s attentional overload. As such, it lowers the user’s frustration caused by receiving too many notifications.

Special properties:

* Focuses on mobile screen use to analyze activity

#### Decision maker [42]:

This concept intercepts notifications from all sources and processes them in a “decision maker” prior to actually arriving at the user. Instead, it processes information from sensors and IoT devices within user and environment contexts to decide upon the target device, type of notification and time of notification. This is done using a machine learning approach. Rather than analyzing the actual patterns in decisions on whether to and how to notify, the paper continues by focusing mostly on the speed and accuracy of various machine learning algorithms.

Special properties:

* Machine learning
* Habit analysis

#### Fuzzy linguistics [28]:

This concept uses fuzzy logic and linguistic variables to analyze the urgency of the reminder and the level of annoyance created by the interruption of the current activity. Resulting from this is a reminder level which determines whether or not the reminder is delayed and/or how the reminder is presented. The urgencies and other levels are all given at design time, however, and are averaged over all users tested prior.

#### PAIR [2]:

This is a relatively older paper which describes one of the first, more advance planners. It takes into consideration several rules as prescribed by the user or caregiver and lays them alongside the activities of the user to provide appropriate reminders. However, no dynamic analysis is done, only design time rules are analyzed.

#### CIA [16]:

Although this paper clearly states “smart reminder”, it doesn’t actually do much in regards to reminding. Instead, it uses image recognition to identify people. After this identification it combines information previously gathered through various systems to display information regarding this person and possible events and reminders tied to them.

Special properties:

* Linking information
* Not directly related to reminders

#### Long term evaluation of smart homes [43]:

Another one not related to reminders per se. This dissertation reviews the users values over long time use of smart home appliances. Their conclusions span generally across all types of smart home appliances. In order for the appliances to provide usefulness it is important that the values of accessibility and trust are upheld. Any appliance which does promote accessibility immediately diminishes any usefulness for the user. Trust generally boils down to the reliability of the provided functionality. If the product still has function impairing bugs, users are likely not to use the product. Even if the producer manages to fix the flaws, the lost trust takes vast time to recover. Another drawn conclusion is that whatever solution implemented, users are initially curious and excited and are willing to try most ideas, but ultimately go back to their routine behavior. As such, the smart appliance should blend into this rather than interrupting it.

#### TEREDA [30]:

Another concept not directly related to reminders. It works by gathering simple data from many sensors around the house and feeding that into the middleware. From this, distributions for the start time and duration are analyzed and used to help recognize activities and cluster them by starting time. For example, there might be 4 clusters of starting times in which the user may generally start to watch TV (with corresponding durations). Each of these clusters may have different subsequent activities, each with different likelihoods. As such, this temporal analysis may be used to predict the likely following activity.

Special properties:

* Activity clustering
* Prediction of next activity

#### What should I do/Action Hierarchies [27], [44]:

These two papers, while again not a directly related to reminders, do portray several underlaying concepts. The first paper presents a framework which represents hierarchical relationships among actions and how values are related to actions. This is formalized in the second paper. Secondly, this framework shows how the relationships tie in with promotion and demotion of values. Lastly, a method is shown on how to infer norms from values rather than vice versa. However, this remains a very theoretical paper.

Special properties:

* Values → Norms
* Actions → Values
* Not directly related to reminders
* Action hierarchy

## Comparison

Below is a comparison of the previously described implementations per the requirements stated in 4.1.

| Concept | RP | RI | SS | Tim | Loc | Act | Env | Dyn | UV | IA | Ref. |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AHCS | x |  |  | x | x | x | x |  |  |  | [24] |
| CAMP | x |  |  |  |  |  |  |  |  |  | [31] |
| CogKnow | x |  | x | x | x | x |  |  | x\* | x | [22] |
| CybreMinder | x |  |  | x | x | x |  |  |  |  | [41] |
| Gate reminder | x |  | x |  |  |  | x |  |  |  | [20] |
| IFTTT | x |  |  | x | x | x | x |  |  |  | [10] |
| MagHive | x |  | x |  |  |  | x | x |  |  | [39] |
| MLCARS | x |  |  |  | x |  |  | x |  |  | [40] |
| Olisto | x |  |  | x | x | x | x |  |  |  | [9] |
| SRS | x |  | x |  | x | x |  | x |  |  | [29] |
| TAFETA | x |  |  | x | x | x | x |  |  |  | [4] |
| Attelia |  | x |  |  |  |  |  |  |  | x | [1] |
| Decision maker |  | x |  | x | x | x | x | x |  |  | [42] |
| Fuzzy lingustics |  | x |  |  |  | x |  |  |  | x | [28] |
| PAIR |  | x |  |  |  | x |  |  |  |  | [2] |
| CIA |  |  |  |  |  |  |  |  |  |  | [16] |
| Goal models |  |  |  |  |  | x |  | x |  |  | [32] |
| HeadacheCoach |  |  |  | x | x | x | x |  |  |  | [35] |
| LTE SH |  |  |  |  |  |  |  |  | x |  | [43] |
| TEREDA |  |  |  | x |  | x |  | x |  |  | [30] |
| WSID/AH |  |  |  | x | x | x | x | x | x |  | [27], [44] |

\* Only at design time

**Legend:**

RP: Reminder Producing

RI: Reminder Intercepting

SS: Specific setup

Tim: Time

Loc: Location

Act: Activity

Env: Environment

Dyn: Dynamic

UV: User Values

IA: Interrupt Analysis

The first fifteen discussed papers and concepts are all smart reminder concepts, whereas the last few describe related concepts such as activity recognition, goal reasoning user values and temporal relations. Overall, the smart reminder concepts can be sub-divided into several groups.

Firstly, the most prominent are those that produce reminders opposed to those that take existing or planned reminders and intercept and process them in some way before actually presenting them to the user.

Secondly, there is a number of concepts which require a specific set-up of hardware opposed to more general, theoretic or software based concepts. These concepts are quite apt and able for those scenarios, but quickly fall short when applied to other scenarios or when generalizing the solution.

The majority of the concepts use (or can use) information about the user or their environment to some extent. Frequently, aside from time, other variables such as location, activity or even weather are used as triggers or conditions for reminders. However, most of these solutions use this information at design time. There are only a few which take it further and use machine learning or other methods in order to create a dynamic system and, for example, predict the subsequent activity and use this information to improve the reminder system. While not mentioned in this section, Timeful [14] would have been an example of this, actually incorporating a self-learning algorithm. However, this application is no longer published and thus cannot be analyzed.

Lastly, user values are not something generally linked to timing smart reminders. To less or more extent, however, they are being used at design time to shape the model.

So, what is useful? There is no existing implementation that can immediately be extended with user values. So in order to arrive at a half-decent concept, rather than extending an existing smart-notification system, it will have to be designed from scratch. However, there are several implementations that contain interesting ideas that can be combined. Most notably [4], [9], [24], [28], [30], [32], [44]. Why these? Of all papers, these show most promise for actual reproducibility and concepts which can be combined.

## User values

The most important aspect is the actual analysis of the appropriate notification moment. A wonderful starting point from within this research group is that of Tielman et al. [44]. Combining their ideas of action hierarchies and possibly that of temporal analysis [30] could lead to very interesting results. Using these concepts to assign value gains and losses to activities allows for quantifying said activities and subject them to further statistical analysis.

If simple identification of one activity which is linked to a goal is not sufficient, usage of the concepts in [32] will allow to identify the necessary activities. This activity or these activities indicate the ultimate deadline before which the reminder should have been presented (i.e. before it’s too late).

Whether or not all or some of the above ideas are used, it is clearly visible that usage of activity information is crucial for prediction and finding useful moments of reminding. Even when exact activity information is not present, the time series of events (even when not activities) is what makes it so interesting and useful for statistical analysis. Especially opposed to other (environmental) variables.

### Which values

For which values should the gains and losses be analyzed? One possibility to look basic human values as proposed by Schwartz [45]. However, these values are very abstract. Govindarajan et al. [46] instead start with 5 basic core value: peace, truth, love, non-violence and right conduct. Subsequently, other values may be derived from this, such as right conduct leading to hygene, punctuality, etc. Looking at other papers that implement values such as [26], [44] similar values are used. As such, there is not really a limit to the values used. As such, in any value related concept, a clear pool of values should be selected.

## Data acquisition

As mentioned previously in section 3.1 no focus is put upon actual activity recognition. As such, this data should be gathered either from existing datasets, from services which provide streaming data, or from existing implementations which use a middleware on top of sensor data to output activity information.

When using raw sensor data, any form of middleware is required before ADL data can be obtained. The first solution is writing such a middleware from scratch. This is the most labor intense solution. However, if the other middleware are not easily implementable or require extensive rewrites, starting from scratch may actually require less work. In [4], they did just that; they designed their own middleware. However, it cannot be used since it remains exactly that, a design. In [24], however, they used an existing middleware [38]. In combination with a context toolkit [47], also used in CybreMinder [41]. The CASanDRA framework [38], however shows great promise since it’s actively used, albeit mostly within its own research group. However, up until the moment of writing this report, it has shown impossible to retrieve its implementation, even after contacting the authors of the original paper and those of papers which used/referenced it.

Rather than using raw sensor data, more labeled data streams may be used. Thank to close ties with the company behind Olisto [9], access is granted to all services and code behind. As such, a simple middleware can easily be built and integrated into their existing infrastructure. Using their information provides direct insight into events (such as device alarms) and states of devices (such as door open or closed) and services (such as weather). This is already an up and running platform, so lots of data is readily available.

Aside from gathering and analyzing data ourselves, the easiest but least extendible option is to use one of the numerous existing data sets scattered over the internet. A select number of these directly provide the desired ADL information.

In [42], dataset [48] was used, but synthetically enhanced to add several properties such as the user activity other than call information and mobile phone usage. As such, it is less interesting in its available, original form.

Three other datasets have been found and are readily available. These, and similar, datasets can be used both for design and for testing. [49], [50] and [51] all have a limited but clear number of activities which are recognized and as such more readily usable. Their differences lie in the number of test subjects and the number of unique activities recorded. Combining data sets is, initially, not a good idea since data points may have different, and thus conflicting, labels. Since the range of activities recorded in these datasets limits the applicable scenarios that can be tested, the most comprehensive data set, [51] is chosen.

## Conclusions

Reflecting on the aforementioned concepts and ideas, we can settle on a few aspects. Firstly, any implementation will have to be built from scratch. Existing implementations are either insufficient or are missing a clear method to reproduce the results. The authors of the different papers have been contacted for their actual implementations (i.e. the code) and have not replied, but with or without them, an implementation should be feasible.

As mentioned previously section 3.1, the assumption is made that a suitable data source is used. Since no feasible middleware for activity recognition exist, a choice is made for an existing dataset. Ergo, there is no direct need for any middleware. A middleware for future connection to any other services that provide streams of data is optional and requires a manual implementation. However, due to close ties with Olisto, an simple middleware may be desired.

The existing concepts and implementations as discussed do not provide any prêt-à-porter solution which can be extended to incorporate user values. Instead, a concept should be designed that builds on the fundamentals of other concepts. Specifically, this means combining these previously mentioned concepts such as activity prediction, linking values to actions and goal reasoning. This actual concept will be explained in the next chapter and will be based on several papers. It will combine several concepts into one model which consecutively can be extended with user values as initially planned.

**Possibly rephrase to answer research questions**

# Concept design

## High-level overview

The initial design is primarily based on combining the ideas of two papers, [30], [44]. In basis, the former paper discusses a method of analyzing data of a user’s ADL and generating a predictive model through a combination two machine learning algorithms: clustering and association rule learning. The latter paper focuses on values and how they link to actions.

Activity prediction

Data acquisition

Model

Data processing

Values

Suggested

notification

Goals

* First, data about user activity is acquired
* Where necessary this data is processed to gain the desired output,
* This data is consecutively processed by the machine learning concepts from [30] to predict subsequent activities.
* This last step is repeated recursively for all subsequent activities, creating a statistical model.
* This model is extended through the concept of values in order to create a statistical model.
* Based on whatever the goal is, the model is used to calculate the ideal moment for notification.

With this basic idea, the model should be able to dynamically adapt to unexpected changes in the user’s behavior. In order to understand this concept fully, first the individual concepts are explained and consecutively the combined design is revisited.

## Processing incoming data

As mentioned before, rather than using raw sensor data and a subsequent middleware, a choice is made to use to use data which directly reflects the user’s ADL. For simplicity and ease of implementation, initially a dataset will be used but options will be kept open for a possible data stream.

### Dataset

The dataset that will be used is one constructed by Sztyler and Carmona [51]. It was chosen for its easy file format, relatively large number of different activities, and the face that it follows more than one person in more than one situation.

As described by Sztyler, “This dataset comprises event logs […] regarding the activities of daily living performed by several individuals. The event logs were derived from sensor data which was collected in different scenarios and represent activities of daily living performed by several individuals. These include e.g., sleeping, meal preparation, and washing. The event logs show the different behavior of people in their own homes but also common patterns.”

Every entry in the dataset simply describes the time of the event, which activity it corresponds to, and whether the event is the start or end of said activity. As such, we need two entries to complete an entry of a single activity.

The format of the dataset is that of XES (Extendable Event Stream) which is an implementation of the XML format. A typical entry for both events of an activity looks like this:



Do note, however, that due to there being separate entries for the start and completion of an activity, it is entirely possible that a second activity may be commenced before the prior one is completed. Furthermore, a complete list of unique activities can be found in Appendix 11.1.

### Data stream

Knowing the format of the dataset which will be used, any format of data stream may be used as long as it contains sufficient information to extract the above properties. In order to do so, a simple middleware will have to be written specifically for the data stream.

Furthermore, when using data streams from Olisto [9], very different types of events may be expected. Unlike activities such as sleeping and washing, activities would include opening and closing the refrigerator or turning off the lights. Whereas this does not directly pose a problem within the concept, it does require a different approach in terms of goal reasoning.

## Activity prediction

Activity prediction is done based on the TEREDA paper by Nazerfet et al. [30]. It focuses on two concepts to create a model for activity prediction; clustering and association rule learning. Clustering is done to improve the accuracy of the prediction model and eliminate possible outliers, while the Expectation Maximization [52] and Apriori [53], [54] algorithms.

### Expectation Maximization

Expectation Maximization (EM) is a clustering algorithm which works iteratively to find maximum likelihood parameters of a statistical model. It is used when such parameters cannot be solved through equations directly. The reason for this may be missing data points, latent variables, or further, still unobserved, data points are to be assumed.

Within clustering there is a division between two types: hard and soft (or fuzzy) clustering. In hard clustering, an element either belongs to a cluster or it does not. In soft clustering, on the other hand, elements can belong to multiple clusters but with different degrees of belief, or confidence. In order to statistically analyze soft clustering, mixture models can be used.

Mixture models are a probabilistically sound way of analyzing soft clustering cases. With this method, each cluster is described as a generative model[[2]](#footnote-2), such as a Gaussian or multinomial. However, the parameters of the model are unknown (for example the mean and covariance in the case of a Gaussian model).

If the source cluster of each observation is known, the estimation of these parameters is trivially done through a simple calculation. However, even when not knowing the source, as is the case in a clustering problem, the EM-algorithm will guess the cluster each point likely belongs to. This is done by using the Baysal formulae, those of conditional probability. However, in order to use these formulae, the parameters of the models need to be known. This leads to a “chicken and egg” problem. The algorithm works on any n-dimensional dataset by first performing a random estimate (expectation) to the initial parameters and iteratively improving (maximizing) them.

In [30], clustering of the activities is done by starting time. Consecutively, outliers are discarded by looking at duration. These clusters are then fed into the Apriori algorithm.

### Apriori algorithm

The Apriori algorithm is a machine learning algorithm used to find patterns in large data sets. Specifically, the patterns of frequent item sets. At its core it attempt to identify frequent item sets in order to generate association rules used to describe general trends in the data. The algorithm finds its roots in analyzing and predicting store transactions to find products frequently bought together.

Every transaction, or customer purchase if looking at the example of a store, is logged in a database. Consequently, a breadth-first search is done to find all items having been purchased at least a percentage of times; the threshold or support. These individual items are extended to larger and larger item sets, given those item sets appear sufficiently often in the database. Using these frequent item data sets, association rules can be generated. The association rules can be described using numerous measures. Among others, there are confidence, lift and conviction [55].

Firstly, the confidence of an association rule indicating X leads to Y, or , is the indication of how often the rule has found to be true. The previously defined support, the indication of how often an item set appears in the data set, can be described as:

**Alle functies moeten een nummer**

Where is a transaction within the database of all transactions . As a result, the confidence of the rule is the proportion of transactions that contain set X, that also contain set Y:

Where is the union of the items in the two sets. Rewritten in probabilities, the support can be seen as simply the probability of an event , where is a transaction containing item set X. However, since regards the items in a set, it can rather be written as . Linking to Bayesian formulae, the confidence can be seen as an estimate of the conditional probability . The drawback of the confidence measure is that it only takes the popularity of itemset X into account.

The lift measure takes both item sets into consideration and compares their dependence to each other to that expected if they were independent of each other. It is defined as:

A lift of 1 would indicate that occurrences of X and Y are independent of each other and thus no rule can be drawn. The higher the value is above 1, the larger the degree in which the occurrence of Y is dependent on that of X and as such is potentially more useful for prediction. Note that a lift below 1 actually indicates that X and Y have a negative impact on each other.

Lastly, the conviction of a rule is an indication of the frequency of an incorrect prediction. It is defined as:

For example, a conviction value of 1.2 indicates that an incorrect prediction occurs 20% more often than if the association was simply by random chance.

The process of the Apriori focuses on first finding all possible datasets which have a minimum support and then creating rules based on the confidence. Depending on the implementation, either just the confidence can be used as a baseline for the rule generation, or a number of measures more. Note that there are more measures of interestingness than just those described above, including, but not limited to, collective strength [56] and leverage [57]. In [30], however, none of the measures other than the confidence are used, which will as such be the starting point for this concept.

The main drawback of the Apriori algorithm is that given the bottom up approach, a large number of subsets are required to be generated. As such, the number of database accesses are very high requiring it to be loaded into memory entirely. Furthermore, the time complexity is obviously very high. Consequently, numerous improved algorithms have been suggested. However, its simplicity makes it much easier to implement on any sort of database. This is interesting because whereas the algorithm is initially only interesting for true transactional databases such as those resulting from stores, the Apriori algorithm can be used to find patterns in any sort of data set.

### Simplification of prediction

In [30], Nazerfad et al.uses the Apriori algorithm to analyze following activities given the cluster of the current activity, as previously found using the EM algorithm. However, there are a few matters that remain unclear in the paper.

Firstly, duration is only used to do away with outliers. However, instead it could instead be very well used within the clustering algorithm itself to produce more accurate results. For now, the procedure of Nazerfad is followed, but this is a very interesting point for future work.

Secondly, Nazerfad suggests the Apriori algorithm but only looks at single step predictions rather than multiple subsequent steps. This simplification is one that is also used in this report since it allows for an easy implementation in a statistical model as will become apparent in 5.5. However, it does discard a great deal of the usefulness of the Apriori algorithm.

## Value based design

Rather than mirroring the paper, like done with activity prediction, reaching a value based smart-reminder systems is done by taking several concepts from [8] and [44]. As explained before, In order for a system to be able to dynamically adapt to the ideas that we deem so logical as humans, the technology needs to have a notion of values.

Thanks to their generalizability and stability over time, values are perfectly suitable for identifying underlying reasons for actions [58]. Formalizing this relationship is complex and can be done in a number of different ways. The simple way used in this report follows that of Tielman [44] and Pasotti [26] in trying to quantify values for computable simplicity: “we propose a simple number which expresses how much an action demotes (negative numbers) or promotes (positive numbers) a value”. Furthermore, the assumption is made that for different actions, the values are commeasurable in order to aid in the computability. However, this assumption “is not a trivial one”.

The logical step would be to go ahead and assign values to all activities in the dataset for further calculation. However, this is not directly useful to the cause. Instead, let’s revisit what we are actually trying to achieve; providing an appropriately time reminder which increases user values.

### The appropriate time

The phrasing of the above question already suggests that an appropriate time for a reminder is one which causes and increase in support for user values. However, if the relationship between actions and their values have been previously quantified, these can be seen as a constant. Instead, focus should be on what effect a reminder has in reducing this value promotion, or even whether it introduces demotion of a certain value. **Wolkje met voorbeeld**

Furthermore, the value gain achieved by the reminder actually having its effect and properly reminding should only count in the calculation when the person actually remembers. Since this is difficult to quantify, an assumption has to be made. **Assuming the person has to be reminded before a critical activity or point in time T, we also assume that having only a short time t between the reminder and the critical point makes the user more likely to remember**. Similarly, being reminded a long time in advance (large t) will have little effect on the person actually remembering.

Combining these two matters, the most appropriate time is the one which shows the larges value gain. This gain is comprised of the effect of the reminder, the quotient introduced by the time between the reminder and the critical moment, and lastly the value loss introduced by the interruption caused by the notification itself. Therefore, we do not actually need information about the activity promoted values themselves, but rather the losses invoked by the notifications and the gains invoked by the reminders. However, these are quantities which have to be taken from the user.

### One value

A difficulty in this is how to compare different values, as well as which values to include at all, as mentioned in 4.4.1. In order for easy statistical analysis it is easiest to look at just a single value. As mentioned in the previous section, the most important aspects are the value losses invoked by the nuisance of the reminder and the gain achieved by actually reminding, which is in turn affected by the time between the reminder and the moment it is too late.

If the gain is taken as a constant, the problem becomes and optimization problem between that number of steps and minimizing the nuisance or invasiveness. The value probably most closely related to this would be emotional well-being, or calmness.

The big advantage of looking at just a single value is that no comparisons have to be made and it is simple for the user to grasp once they are expected to provide feedback regarding this value.

### How to measure?

Ideally a program or model would make an initial estimate based on previous users and adjust its parameters based on this user’s input and past activities. However, such an implementation would invoke a drastic increase of scope and require much more research, programming and testing.

## Statistical analysis

Recapping on the current status, using the clustering and prediction techniques, given any current activity we can predict a next activity with a certain probability. This, however, only brings us one step further in time. Remember that the aim is to find an appropriate moment for a notification.

**Wolkje:**

*Describing a goal in a statistical model is a difficult feat. Usually, reaching a goal comprises of doing a few subsequent activities. That is where goal reasoning [32] comes into play. However, in simple scenarios, such as Peter remembering to close the garden doors before going to sleep or leaving the house, we can easily link goals directly to an activity. In this specific case, we want to send a reminder before activity ‘sleep’, given ‘closing doors’ has not yet been performed. Now the latter part can easily be checked, so the focus is on reminding before the deadline activity, ‘sleep’. While it is possible to let the statistical model dynamically adapt to having a single or more of such goal activities, for simplicity, only a single goal activity is accepted.*

For simplicity, we accept that reaching the goal is synonymous to reaching a certain activity. As such, the notification should be presented before this activity, but not too long before. The problem can as such be illustrated as:

Given a current activity A and a goal activity Z, we are looking for an activity S with the highest value, that will be reached with only a minimal number of expected steps remaining before we reach Z. So:

**A → [n steps] → S → [m steps] → Z**

Where the aim is to find a minimal m with a maximal value for S.

Now one way would be to simply traverse the probability tree, one activity after another. However, this is a very intensive process. Assuming there are enough recorded activities, we can safely **assume** **(wolkje)** that at one point the user reaches a similar activity to one performed before. As a result, the activities can no longer be modeled just as a tree, but also as a (discrete time) Markov chain where every activity is a state. The advantage of this is that there are numerous documented ways to analyze such chains. However, in order to do that, we first need to more clearly define the problem mathematically. This is done by analyzing the two components that need to be found: the expected value of S and a corresponding expected number of steps from S to Z. These two steps can both be calculated through the use of Markov chains.

**Ergens wolkje met uitleg wat markov chain precies is… blabla over FSM:**

“A Markov chain is a mathematical process that transitions from one state to another within a finite number of possible states. It is a collection of different states and probabilities of a variable, where its future condition or state is substantially dependent on its immediate previous state. These probabilities can be exhibited in the form of a transition matrix.” [59], [60]

**Illustratie probability tree naar markov chain**

### Expected value

The expected value of any random variable X can be defined as the probability-weighted average of all possible values X can take on [61]. Accepting that any activity is linked to a certain number describing its value, this can be written as:

All that remains is to find the probability of each state. Given that we know all transient probabilities, a transition matrix of the system can be built. As an example, we will take a system of three states that represent three activities as shown in the figure below.



Figure 5.1: Example state model with values

In the model it can be seen that each state has a certain value and a probability to reach a different state next. These probabilities and values can be combined in a so called transition matrix and a value matrix. For the example model that would result in respectively:

The use of the transition matrix is plentiful. For example, taking describes the probabilities of reaching any state, given a starting state, after 3 steps.

**Definitiewolkjes**

**Irreducible: Ability to get from any state to any state**

As mentioned before, an important characteristic of our system is that we are able to reach any state from any state since it is a repeating system. In simple words, at any moment we know that a person, at one point, will fall asleep again. This property allows finding the stationary probabilities; the steady-state probabilities as the number of steps taken approaches infinity. As such, the equation we are attempting to solve is:

Where is a row vector whose entries are the probabilities of the states, all summing to 1. For a small number of states, this can be manually done using some simple variable substitution and some linear algebra knowledge. Especially for larger systems, however, working with the full matrices is easier. With some quick rewriting:

This can be solved by finding the eigenvalues and corresponding eigenvectors of the matrix . Of course, there will be multiple eigenvectors, but the one corresponding to the stationary distribution is that for which all entries of the eigenvector are positive. In our example, this would be the eigenvector corresponding to the eigenvalue , which is . Consequently, the stationary distribution is:

Having found the stationary distribution, the corresponding expected values are only a step away:

As shown, this is a simple solution to finding probabilities and expected values for an irreducible Markov chain. There is, however, a problem when we look at the problem posed at the beginning of this section. When going from state A, through S, to final state Z we require state S to be reached before the final state Z. The model that was just discussed doesn’t see state Z as a final or ‘losing’ state but simply allows for continuation. In order to solve this problem, the transition matrix has to undergo a transformation and make the final state absorbing.

### Absorbing Markov chain

**Definitie: Absorbing state: a state that, once entered, cannot be exited.**

**Recurrent state: a state that, once exited, can always be returned to**

**Transient state: a state that, once exited, cannot always be returned to**

An absorbing Markov chain is a chain where one or multiple states are absorbing and thus cannot be left, while all other states can reach at least one of these absorbing states [62]. Like with normal Markov chains, the transition matrix can be used to calculate a number of interesting properties.

Most notably, since the system is no longer irreducible, its steady-state distribution has changed. As the number of steps lean towards infinity, the probability will always be 1 for all absorbing states combined and 0 for all other transient states. This might seem like it’s more difficult to work with, but in actuality it can be approached in a similar way as a converging series. Through some simple transformations and easy calculations, it is actually possible to calculate the expected number of steps between any state and an absorbing state as well as other interesting properties.

So where to begin? First and foremost, the system has to be made absorbing.

**Redo state system**

Taking state C as our finals state, our new, absorbing, transition matrix is as follows:

It is nothing other than making the corresponding element equal one while all other elements in the row are reduced to zero. To continue in the process of attaining the interesting aspects of our system, the fundamental matrix is needed. In order to find this matrix, several components are needed first. These components can be gathered from the new transition matrix once it is written in canonical form. For a transition matrix with t transient states and r absorbing states, the canonical form is described as:

: a *t-by-t* matrix, describing probabilities from a transient state to another

: a nonzero *t-by-r* matrix describing probabilities from a transient state to an absorbing states

**0**: an *r-by-t­* matrix of zeros

: an *r-by-r* identity matrix

In our example case, the transition matrix is already in canonical form so the property matrices can be obtained immediately:

The fundamental matrix is a matrix that describes the expected number of visits of a transient state before being absorbed. In order to find that, we start with the property matrix . Entry (*i,j*) of describes the probability of going from state *i* to state *j* in exactly one step. The same entry in describes the probability in exactly two steps, etc. As mentioned before in 5.5.1, the expected value of anything is calculated by summing all probabilities multiplied by their values. Doing exactly that for each entry yields our fundamental matrix:

Calculating that for the example, we find:

Here, entry (*i,j*) of describes the expected number of times the Markov chain is in state , given that the chain starts in state , before being absorbed. From this, the expected number of steps before being absorbed can be calculated with a simple formula:

Here, **1** is a column vector of the same dimension as N containing all ones. Calculating this for our example, we find:

So we know it will take an average of 1.34 or 1.13 steps before being absorbed if started in state A or B respectively.

While it is great that we now have the number of steps before being absorbed, the previous method of calculating the probability of reaching any one state before being absorbed can no longer be used. Luckily, also for absorbing Markov chains there is a way to calculate this:

**Definitiewolkje: Diagonal matrix: a matrix in which all entries outside the main diagonal are 0**

Where is a diagonal matrix with the same diagonal as N. The resulting matrix H describes the probability of visiting any transient state given a starting state. Calculating that for our example, we find:

In order to now achieve the expected values, we simply multiply every element in each row by their values:

Recapping on the original problem, the goal was to find a state S such that in:

**A → [n steps] → S → [m steps] → Z**

We achieve a maximum expected value for S and a minimum number of steps for m. In order to find the expected value for S, equation (5.13) is used to find the probabilities of each state given the current starting state and then multiplied by the values of each possible state S. Consequently, the expected number of steps is found through equation (5.12). Looking at the example for the last time, taking A as our current state and C as our final state, we find for our intermediate state B:

For both *T* and *H*, the results are quite obvious for such a small system. However, imagine that for this to work in a system of a user’s ADL large number of activity clusters that this is no longer an easy feat. Since we are trying to optimize the expected value for S while keeping the number of steps until absorption at a minimum, these calculations have to be done for all possible S. After this, a decision can be made through a weighting formula. Thankfully, the matrix calculations performed are basic ones that have plentiful implementations in numerous programming languages as will be seen further on in the report.

**In apart ding:** Note that when calculating the number of steps that in order to find m, state S is used as a starting state instead of state A. The reason for why we can do this is that we are assuming that the Markov chain is time-independent; that the probabilities don’t change as steps are taken. Of course in real life, probabilities will change as new activities are being recorded. However, with a sufficiently large number of existing records, these changes in probability are minute. Furthermore, this will be calculated after every recorded activity. As such, at that moment, the system is indeed time-invariant.

### Drawback of choosing Markov chains

There is one big drawback of using Markov chains. This assumes complete independence between past and future states other than just the one step. In reality, however, there are usually a series of sequential activities. For now this is not taken account. However a possible solution would be to look at every possible set resulting from the Apriori algorithm and map is as a separate state. Even this is not ideal but it would be an improvement.

## Concept description

Now that all aspects of the concept have been discussed, we can revisit the combined design and see where any changes might be needed. Figure XXX shows a more detailed overview of the earlier posed design.

Data acquisition & processing

First, data about user activity is acquired either from a stream or a dataset. For now a data set is used, but the possibility of a data stream is accounted for. The data is normalized to contain type, starting time and duration.

Activity prediction

The data is clustered using an Expectation Maximization algorithm. A prediction for each next cluster is made using the Apriori algorithm.

Values

Rather than looking at all possible user values, only the nuisance caused by the notification is regarded. Through manual input the user’s values for every activity are stored.

Model

All cluster predictions are combined and modelled as a Markov chain with corresponding transition matrix

Goal

The goal activity, before which the reminder has to be dispatched to the user, should be provided. For now this is done manually and only a single such activity is selected. Since the cluster of this goal is irrelevant, these clusters are combined in the model.

Moment selection

The model is made absorbing, based on the provided goal activity. Using the properties of an absorbing Markov chain, for each cluster, the following properties are calculated:

* Probability of reaching that state/cluster before being absorbed (reaching the deadline activity)
* The corresponding expected value as calculated from the probability and the value as supplied by the user
* The expected number of steps until absorption is calculated

Subsequently, the expected value can be optimized with respect to the expected number of steps. This optimization is done through a formula which can be tweaked based on the achieved results. Whatever cluster is ranked highest will be selected for the planned notification.

Every time a new activity is recorded, its cluster calculated and the model is updated. Whenever the current activity cluster corresponds with the highest ranked one, the notification is dispatched. If no timely notification is planned and the goal activity is reached, the notification should be dispatched immediately.

**Dit even controleren of het na het testen nog exact zo gaat**

Activity prediction

Data acquisition

Model

Data processing

Values

Moment selection

Goals

Dataset

Data stream

EM

Apriori

Loss by notification

Markov chains

Goal activity identification

Dynamic model checking

# Implementation

The implementation is a major aspect in this report. In order to test the proposed concept, a number of things have to be done. First, a suitable platform has to be chosen. This platform should not only allow for all desired datasets to be supported, but preferably also allow for connection to a real-life application for future field testing. Secondly, the algorithms of the conceptual design have to be implemented in code and linked to one another and to the data sources. Lastly, the implementation should allow for some sort of reporting mechanism which allows analysis of the results.

## Platform

What platform to choose isn’t just dependent on what algorithm is chosen, or what libraries are available. More important is to see how the data is obtained. Keeping an open mind as to where data can come from, and not just restricting oneself to using premade datasets, allowing streaming data is important. Why? Because of the rapid rise in Internet of Things devices.

### Internet of Things

The field of activity recognition is a rapidly evolving one. This is mainly due to the exponential rise in Internet of Things (IoT) devices. Currently, there are over 17 billion connected devices in the world. Of these, there are over 7 billion IoT devices (so excluding smartphones, computers and similar) with over 6.5 million new devices being connected every day [37]. This is expected to grow to between 20 and 200 billion within the next five to ten years. The promise of IoT doesn’t end at just connecting the devices to the internet. It is just the first step.

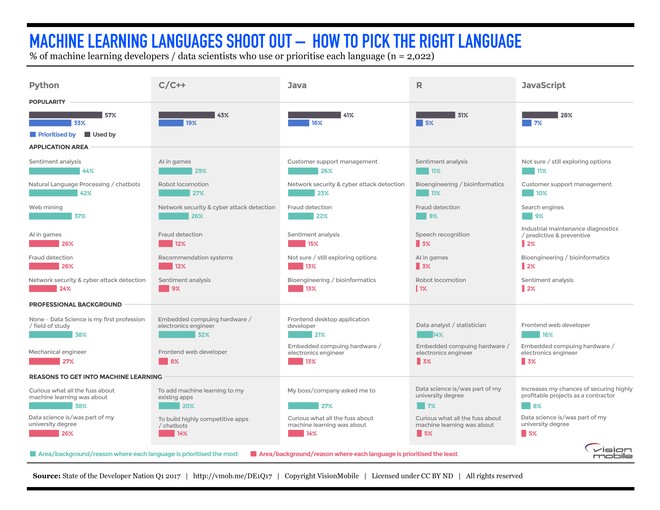
Advances in RF technology and low power computing will bring Internet-connectivity everywhere. Advances in Big Data and machine learning will unlock new business opportunities and models. The possibilities are nearly endless, but they all still lie quite out of reach from the direct consumer. However, specifically for activity recognition, suddenly a lot more data is available than there was 10 years ago. Consequently, more and more papers and implementations, such as [63]–[65], are analyzing activity based on random sensor data.

Whether the activity data or the sensor data is available, in any case a prediction can be made on past events. As long as the event corresponding to the deadline is known before which the notification should have been presented, any form of data should fit within the design. As such, a server based solution, preferably in the cloud, seems most logical.

### Programming language

When it comes to implementing machine learning algorithms, there are several go to languages. The five most used languages [66], in order, are:

* Python
* C/C++
* Java
* R
* JavaScript



**Do something with this figure for a nice visual**

While there are many other options, they fall below a 5% mark of prioritization in the field of machine learning. Python takes the clear lead in this field. This is due to the large number of readily available libraries. This dramatically decreases the time required to implement machine learning algorithms in applications. However, regardless of popularity it is shown that professional background is key to choosing a language.

For now ignoring the fact of whether the programmer has any existing proficiencies, it is important to note that there is no best language to use for machine learning and it is important to take the goal into consideration. In this case the goal is to create a server based cloud platform. Whereas the algorithms can still be run on any language, the web part and a possible API interface are likely to be implemented in JavaScript.

## Setup

Taking the above choices into consideration and looking at the current professional landscape, there is a single, simple way forward. There are two reasons for this. First, almost all web based APIs work using HTTP requests. As such, a setup is needed which can perform all calculations as well as communicate via HTTP requests. Second, when considering a JavaScript based platform, the largest market share (over 60% [67]) is attributed to Node.js webservers.

### Software platform – Node.js

**Misschien als wolkje**

JavaScript was originally a client-side scripting language, running in the user’s browser, usually part of any website. Node.js [68] changed the game by providing an open source platform allowing any JavaScript based application to run outside of a browser. It’s main advantage for programmers is that only a single language would have to be used for both frontend and backend (client-side and server-side) implementations.

### Software library – npm

Aside from the above, an important feature of Node.js is that it has an expansive repository of packages that can be imported for use in applications. This Node.js package manager (npm) [69] is embedded within Node.js and as such packages can be accessed as libraries, directly from the code. In order to achieve all desired functionalities, without reinventing the wheel, several important packages are used and described below.

#### Express

Express [70] is a framework that facilitates and simplifies the creation of web applications and services. It is built over the native HTTP module within Node.js and allows for much quicker implementations of such functionalities. Most notably, it simplifies routing when used in conjunction with an API or website.

**Ergens wolkje:**

**Routing: Routing is handling incoming http requests from a client. Based on the route that was accessed, a different code will be executed. As an example, doing a Google search is basically doing a request to fetch data (a GET request) from the route ‘/search/’, with the host ‘google.com’, along with a number of parameters. More on routing will be explained in 7.2.**

#### Mongoose

Mongoose [71] provides a straight-forward, schema-based solution to model application data as it is stored in a MongoDB type database (described later in 6.2.3). It simplifies query building and handles type casting. Based on the schemas, it allows creating model objects that are synonymous to a table entry in the database. Subsequently, all creations, deletions and edits are simplified.

#### ml.js

The ml.js suite [72] is a series of machine learning related libraries written in JavaScript. Most notable are the inclusions of tools for complex matrix calculations (for Markov chain analysis), as well as clustering and predictions. As such, it contains all tools required to perform the calculations and analysis as described in chapter 5.

### View engine – Handlebars

The application needs to, among other things, handle user input and allow for data sets to be imported. For this, the simplest solution is to do all user interaction through the means of a webpage. While Node.js can natively serve html back to the client-side upon request, hardcoding the entire layout into every page is tedious work. Using a view engine allows the programmer to work according to templates where content is filled in according to a route. This allows views (the visuals) and code to be separated. Handlebars [73] is such a templating engine. While each engine has its advantages and functions, Handlebars is one of the most minimalistic. Since no complex views are required a minimalistic approach is preferred.

### Database – MongoDB

In order to store data regarding activities, clusters and users, a database is required. While there are plentiful options when it comes to databases that work with Node.js, there is one big advantage to using MongoDB [74]. It allows handling unstructured data. Typically, a database requires a clearly defined structure, and works with rows in a table. MongoDB, instead, works with documents. These documents are described by a schema, such a schema can still be vague.

The direct consequence of such a system is that no initial thought has to be put into the structure of the database and it can be structured on-the-fly. This greatly reduces workload early in the programming process, allowing for more time spent on the actual implementation. Throughout the process of the implementation, the databased can be remodeled and optimized upon new findings. In more traditional databases, this is not always as easy. Although arguments can be made that requiring more planning upfront ultimately leads to a better structured, and thus a more optimized, database, this is not the current desire.



**Ergens blabla over ml-matrix en doe iets over scaling**

**Ergens plaatje van de Notival site**

# System architecture

## Schematics

Drawing of entire system architecture

**When done**

## API

Bla bla over het maken van een restful systeem

Description of all endpoints and what they do

**When done**

# Experimentation

## Testing

## Results

# Discussion

* Future: wel onderscheid maken tussen values
* Verschil importance verschillende waarden
* Wel kijken naar bonus van waarde als reminder doorgaat want mss is de reminder wel helemaal niet beneficial
* Kijken naar duration als onderdeel van clustering
* Daadwerkelijk meenemen van aprior sets
* Mogelijkheid tot absorption verschillende activiteiten

# References

[1] T. Okoshi, H. Nozaki, J. Nakazawa, H. Tokuda, J. Ramos, and A. K. Dey, “Towards attention-aware adaptive notification on smart phones,” *Pervasive Mob. Comput.*, vol. 26, pp. 17–34, Feb. 2016.

[2] L. S. Shafti, P. A. Haya, M. García-Herranz, and X. Alamán, “Personal Ambient Intelligent Reminder for People with Cognitive Disabilities,” in *Ambient Assisted Living and Home Care*, 2012, pp. 383–390.

[3] J. K. Zao, M. Y. Wang, P. Tsai, and J. W. S. Liu, “Smart phone based medicine in-take scheduler, reminder and monitor,” in *The 12th IEEE International Conference on e-Health Networking, Applications and Services*, 2010, pp. 162–168.

[4] A. Arcelus, M. H. Jones, R. Goubran, and F. Knoefel, “Integration of Smart Home Technologies in a Health Monitoring System for the Elderly,” in *21st International Conference on Advanced Information Networking and Applications Workshops, 2007, AINAW ’07*, 2007, vol. 2, pp. 820–825.

[5] W. Jih, J. Y. Hsu, and T.-M. Tsai, “Context-Aware Service Integration for Elderly Care in A Smart Environment,” 2006.

[6] N. Mitabe and N. Shinomiya, “Support system for elderly care with ambient sensors in indoor environment,” in *2017 Eleventh International Conference on Sensing Technology (ICST)*, 2017, pp. 1–4.

[7] M. Neerincx, M. Tielman, C. Horsch, W.-P. Brinkman, K. Bosch, and R. J. Beun, “Virtual Health Agents,” 2015.

[8] M. B. van Riemsdijk, C. M. Jonker, and V. Lesser, “Creating Socially Adaptive Electronic Partners: Interaction, Reasoning and Ethical Challenges,” in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, Richland, SC, 2015, pp. 1201–1206.

[9] “Olisto makes smart thing smarter, according to your rules.,” *Olisto*. [Online]. Available: https://olisto.com/. [Accessed: 19-Apr-2018].

[10] IFTTT, “IFTTT helps your apps and devices work together.” [Online]. Available: https://ifttt.com. [Accessed: 19-Apr-2018].

[11] “Maps - Navigation & Transit - Apps on Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.maps&hl=en. [Accessed: 19-Apr-2018].

[12] “Free Community-based GPS, Maps & Traffic Navigation App | Waze.” [Online]. Available: https://www.waze.com/en. [Accessed: 19-Apr-2018].

[13] Peter G. Chin, “Smart reminders,” 25-Mar-2005.

[14] “Timeful,” *Internet Archive*, 02-Mar-2015. [Online]. Available: https://web.archive.org/web/20150302091124/http://www.timeful.com/. [Accessed: 19-Apr-2018].

[15] N. Clarkson, “Spotify to predict the music you want to listen to,” *Virgin*, 19-Nov-2014. [Online]. Available: https://www.virgin.com/music/spotify-to-predict-the-music-you-want-to-listen-to. [Accessed: 19-Apr-2018].

[16] F. Kargl, B. Dong, T. Illmann, and M. Weber, *Smart Reminder - Personal Assistance in a Mobile Computing Environment*. 2002.

[17] A. Robertson, “Location/time-based reminder for personal electronic devices,” 06-Dec-2000.

[18] Jason F. Hunzinger, “Location specific reminders for wireless mobiles,” 15-Nov-2001.

[19] Michael Sean McGee, Michael S. McIntyre, and James Randall Walker, “Generating an alarm based on location and time,” 17-Apr-2003.

[20] S. W. Kim, M. C. Kim, S. H. Park, Y. K. Jin, and W. S. Choi, “Gate Reminder: A Design Case of a Smart Reminder,” in *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, New York, NY, USA, 2004, pp. 81–90.

[21] S. Helal, C. Giraldo, Y. Kaddoura, C. Lee, H. El Zabadani, and W. Mann, “Smart Phone Based Cognitive Assistant,” Apr. 2018.

[22] D. Zhang, M. Hariz, and M. Mokhtari, “Assisting Elders with Mild Dementia Staying at Home,” in *2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2008, pp. 692–697.

[23] M. Philipose *et al.*, “Inferring activities from interactions with objects,” *IEEE Pervasive Comput.*, vol. 3, no. 4, pp. 50–57, Oct. 2004.

[24] A. Hristova, A. M. Bernardos, and J. R. Casar, “Context-aware services for ambient assisted living: A case-study,” in *2008 First International Symposium on Applied Sciences on Biomedical and Communication Technologies*, 2008, pp. 1–5.

[25] M. S. Kließ and M. B. van Riemsdijk, “Requirements for a Temporal Logic of Daily Activities for Supportive Technology.”

[26] P. Pasotti, M. B. van Riemsdijk, and C. M. Jonker, “Representing human habits: towards a habit support agent,” in *Proceedings of the 10th International workshop on Normative Multiagent Systems (NorMAS’16)*, 2016.

[27] P. Pasotti, C. M. Jonker, and M. B. van Riemsdijk, “Towards a formalisation of Action Identiﬁcation Hierarchies∗.”

[28] S. Zhou, C.-H. Chu, Z. Yu, and J. Kim, “A context-aware reminder system for elders based on fuzzy linguistic approach,” *Expert Syst. Appl.*, vol. 39, no. 10, pp. 9411–9419, Aug. 2012.

[29] H. T. Chaminda, V. Klyuev, and K. Naruse, “A smart reminder system for complex human activities,” in *2012 14th International Conference on Advanced Communication Technology (ICACT)*, 2012, pp. 235–240.

[30] E. Nazerfard, P. Rashidi, and D. J. Cook, “Using Association Rule Mining to Discover Temporal Relations of Daily Activities,” in *Toward Useful Services for Elderly and People with Disabilities*, 2011, pp. 49–56.

[31] S. Vurgun, M. Philipose, and M. Pavel, “A Statistical Reasoning System for Medication Prompting,” in *UbiComp 2007: Ubiquitous Computing*, 2007, pp. 1–18.

[32] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, “Reasoning with Goal Models,” in *Conceptual Modeling — ER 2002*, 2002, pp. 167–181.

[33] H. Oinas-Kukkonen, “A foundation for the study of behavior change support systems,” *Pers. Ubiquitous Comput.*, vol. 17, no. 6, pp. 1223–1235, Aug. 2013.

[34] R. Klaassen, “HCI Perspectives on Behavior Change Support Systems,” Feb. 2015.

[35] A. Fritzen, N. Leipold, N. Terzimehic, M. Böhm, and H. Krcmar, “HeadacheCoach: Towards Headache Prevention by Sensing and Making Sense of Personal Lifestyle Data,” 2017.

[36] E. S. Poole, “HCI and mobile health interventions,” *Transl. Behav. Med.*, vol. 3, no. 4, pp. 402–405, Dec. 2013.

[37] “State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating.” .

[38] A. M. Bernardos, P. Tarrío, and J. R. Casar, “CASanDRA: A Framework to Provide Context Acquisition Services ANd Reasoning Algorithms for Ambient Intelligence Applications,” in *2009 International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2009, pp. 372–377.

[39] “MagHive - World’s First Modular Smart Reminder (Canceled),” *Kickstarter*. [Online]. Available: https://www.kickstarter.com/projects/2034560442/maghive-worlds-first-modular-smart-reminder. [Accessed: 24-Jul-2018].

[40] P. Shanahan, “Machine Learning for Context-aware Reminders and Suggestions,” PhD Thesis, University of California at San Diego, La Jolla, CA, USA, 2009.

[41] A. K. Dey and G. D. Abowd, “CybreMinder: A Context-Aware System for Supporting Reminders,” in *Handheld and Ubiquitous Computing*, 2000, pp. 172–186.

[42] F. Corno, L. D. Russis, and T. Montanaro, “A context and user aware smart notification system,” in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015, pp. 645–651.

[43] G. Sandström and Kungliga tekniska högskolan (Stockholm), *Smart homes and user values: long-term evaluation of IT-services in residential and single family dwellings*. Stockholm: Royal Institute of Technology, 2009.

[44] M. L. Tielman and C. M. Jonker, “What should I do? Deriving norms from actions,values and context,” p. 5.

[45] S. H. Schwartz, “An Overview of the Schwartz Theory of Basic Values,” *Online Read. Psychol. Cult.*, vol. 2, no. 1, Dec. 2012.

[46] M. Govindarajan, S. Natarajan, and V. S. Senthilkumar, *Professional Ethics and Human Values*. PHI Learning Pvt. Ltd., 2013.

[47] D. Salber, A. K. Dey, and G. D. Abowd, “The Context Toolkit: Aiding the Development of Context-enabled Applications,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 1999, pp. 434–441.

[48] N. Eagle and A. (Sandy) Pentland, “Reality mining: sensing complex social systems,” *Pers. Ubiquitous Comput.*, vol. 10, no. 4, pp. 255–268, May 2006.

[49] “Activity Recognition Challenge Dataset Download | Opportunity.” [Online]. Available: http://www.opportunity-project.eu/challengedatasetdownload. [Accessed: 03-Aug-2018].

[50] F. J. Ordóñez, P. de Toledo, and A. Sanchis, “Activity Recognition Using Hybrid Generative/Discriminative Models on Home Environments Using Binary Sensors,” *Sensors*, vol. 13, no. 5, pp. 5460–5477, Apr. 2013.

[51] Sztyler, T. (Timo) and Carmona, J. (Josep), “Activities of daily living of several individuals.” University of Mannheim, Germany, 2015.

[52] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *J. R. Stat. Soc. Ser. B Methodol.*, vol. 39, no. 1, pp. 1–38, 1977.

[53] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules in Large Databases,” in *Proceedings of the 20th International Conference on Very Large Data Bases*, San Francisco, CA, USA, 1994, pp. 487–499.

[54] R. Agrawal, T. Imielinski, A. Swami, H. Road, and S. Jose, “Mining Association Rules between Sets of Items in Large Databases,” p. 10.

[55] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, *Dynamic Itemset Counting and Implication Rules for Market Basket Data*. 1997.

[56] C. C. Aggarwal and P. S. Yu, “A New Framework for Itemset Generation,” in *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, New York, NY, USA, 1998, pp. 18–24.

[57] G. Piatetsky-Shapiro, “Discovery, Analysis, and Presentation of Strong Rules,” in *Knowledge Discovery in Databases*, 1991.

[58] S. H. Schwartz, “Universals in the content and structure of values: theoretical advances and empirical tests in 20 countries,” in *Advances in Experimental Social Psychology*, 1992.

[59] “What is a Markov Chain? - Definition from Techopedia,” *Techopedia.com*. [Online]. Available: https://www.techopedia.com/definition/8249/markov-chain. [Accessed: 11-Jan-2019].

[60] J. G. Kemeny, *Finite Mathematical Structures*. Prentice-Hall, 1960.

[61] S. M. Ross, *Introduction to Probability Models*. Academic Press, 2007.

[62] J. G. Kemény and J. L. Snell, *Finite markov chains*. Van Nostrand, 1960.

[63] W. D’Almeida, “Deep learning for sensor-based human activity recognition,” *Becoming Human: Artificial Intelligence Magazine*, 05-Jan-2018. [Online]. Available: https://becominghuman.ai/deep-learning-for-sensor-based-human-activity-recognition-970ff47c6b6b. [Accessed: 12-Jan-2019].

[64] N. C. Krishnan and D. J. Cook, “Activity recognition on streaming sensor data,” *Pervasive Mob. Comput.*, vol. 10, pp. 138–154, Feb. 2014.

[65] A. Jordao, A. C. Nazare Jr., J. Sena, and W. R. Schwartz, “Human Activity Recognition Based on Wearable Sensor Data: A Standardization of the State-of-the-Art,” *ArXiv180605226 Cs*, Jun. 2018.

[66] M. Wilcox, S. Schuermans, C. Voskoglou, and A. Sobolevski, “State of the Developer Nation,” 2017.

[67] “Usage Statistics and Market Share of Node.js for Websites, January 2019.” [Online]. Available: https://w3techs.com/technologies/details/ws-nodejs/all/all. [Accessed: 12-Jan-2019].

[68] N. js Foundation, “Node.js,” *Node.js*. [Online]. Available: https://nodejs.org/en/. [Accessed: 12-Jan-2019].

[69] “npm.” [Online]. Available: https://www.npmjs.com/. [Accessed: 12-Jan-2019].

[70] “Express - Node.js web application framework.” [Online]. Available: https://expressjs.com/. [Accessed: 12-Jan-2019].

[71] “Mongoose ODM v5.4.3.” [Online]. Available: https://mongoosejs.com/. [Accessed: 12-Jan-2019].

[72] “mljs/ml,” *GitHub*. [Online]. Available: https://github.com/mljs/ml. [Accessed: 12-Jan-2019].

[73] “Handlebars.js: Minimal Templating on Steroids.” [Online]. Available: https://handlebarsjs.com/. [Accessed: 12-Jan-2019].

[74] “Open Source Document Database,” *MongoDB*. [Online]. Available: https://www.mongodb.com/index. [Accessed: 12-Jan-2019].

# Appendices

## Unique activities in dataset

**ToDo**

1. A virtual geographic boundary, defined by GPS or RFID technology, that enables software to trigger a response when a mobile device enters or leaves a particular area. [↑](#footnote-ref-1)
2. In machine learning (and other forms of statistical classification) there are two main approaches: generative and discriminative. Given a target Y and an observation X, the generative model is a statistical model of the joint probability distribution. Whereas the discriminative model looks at conditional probability of Y given X=x. [↑](#footnote-ref-2)