NotiVal

Value-Based Smart Reminders

Finding appropriate moments for notifications in smart reminder system

Draft
Remy Kabel
14/01/2019

Master Thesis R. Kabel
MSc Embedded Systems | Interactive Intelligence
TU Delft

User valued smart reminders

Finding appropriate moments for support in socially adaptive electronic partners

Opmerkingen
- Verwijs naar figuren in tekst
- Sowieso nooit verwijzen naar figure/tabellen zonder nummer. Dus geen: “Below”, “On the right” etc!

- Check figuur verwijzingen en beschrijvingen.

- Voorblad maken

- Pagina hiernaast kloppend maken

Student

R. Kabel (*4132165*)
Msc Embedded Systems
TU Delft

Supervisory team

Dr. Willem-Paul Brinkman (chair)
Department: Interactive Intelligence

Dr. M. Tielman (mentor)
Department: Interactive Intelligence

TU Delft

Abstract

This project focuses on finding what defines an appropriate moment to notify in a smart re-minder system. Specifically, the goal is to find a way in which smart reminders systems can be extended through the use of user values to ultimately provide appropriately timed supportive feedback. A system is designed from scratch, combining existing concepts of activity prediction and value-based design. A statistical Markov chain model is made from predictions based on Expectation Maximization and Apriori algorithms. User values are quantified and optimized following the concept of a Socially Adaptive Electronic Partner and added to the model to identify an appropriate moment for a notification. The model is implemented in a Node.js web application, following the principles of a RESTful web API. The statistical model shows a 91% success rate, but falls short at 73% when considering values. Up to a 13% improvement in the success rate and an 18% improvement in the score (more appropriate moment) is found for the model considering user values with respect to basic, predictive model.. Overall, a clear approach to value-based smart reminders is shown in a statistical and dynamic approach to incorporate the concept of user values in a smart-reminder system.

Table of contents

Table of common terms

Term	Description
ADL	Activities of daily living
SAEP	Socially Adaptive Electronic Partner
Middle wear	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	An endpoint (or address) for an HTTP request
Hostname	Label or address used to identify a device. Usually this will be the do-main linked to a certain IP address. For example: google.com
Endpoint/URL	Universal resource locator. The location, or address, of a certain resource. For example: http://www.google.com/search?query=blah
Path	The location identifying component of the URL. For http://www.google.com/, this would be '/'. For http://www.google.com/search?query=blah, this would be '/search'
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

Acknowledgements

First and foremost, I would like to thank Dr. Birna van Riemsdijk for her guidance in the first part of my thesis process. I wish her lots of health and a great time at her new position in Twente.

Aside of this, an equally big thank you to Willem-Paul Brinkman for taking over her role as supervisor and providing clear and decisive support.

I would like to thank my boyfriend and parents and roommate for their never-ending support. Piotr, Michel, Jacqueline, and Joanna, I love you all!

A humongous hug to Marit van de Kamp for being my InDesign champion and go-to listening ear.

Finally, a big hug and many butt squeezes to my two amazing dance partners, Monique Baijer-Belde and Ilona Schooneveld who have been with me through it all. You two are amazing!

Reader context

This report was written for a diverse audience. It is equally meant for academics from various fields, experts from related companies, as well as the intelligent, common reader. As such, a choice was made to include explanations of several basic 'textbook' principles of the various topics.

1. Introduction

It is natural for humans to increase their dependence on technology as technology improves [1]. Through applications in smart homes, wearables, virtual coaches and many others, people have increasingly adapted modern technologies into their daily lives. Their goals are to increase health, efficiency, and many other values. Conversely, the abundance of apps and notifications have a negative effect on the users and cause them to grow immune to the constant stream of information that is presented to them on a daily basis [2]. Especially the elderly or people with a mental impairment could benefit from an effective support agent [3]–[8].

A proper implementation of such a support agent could improve the effectiveness of all notification-based applications. "Too many notifications cause the user to tune out" [9]. Rather than bombarding the user with notifications right when a related event occurs, the user is much more likely to act upon the notification if it is delivered at an appropriate time. But what actually is an appropriate time?

The appropriate time for notification 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.*

In this example, timely notification is of the essence. The current situation? Usually, a simple timer is set moments before the expected time of the deadline. Obviously, this is not very foolproof. The ideal solution? Getting a notification just before the deadline of sleeping or leaving the house. A smart reminder, so to say.

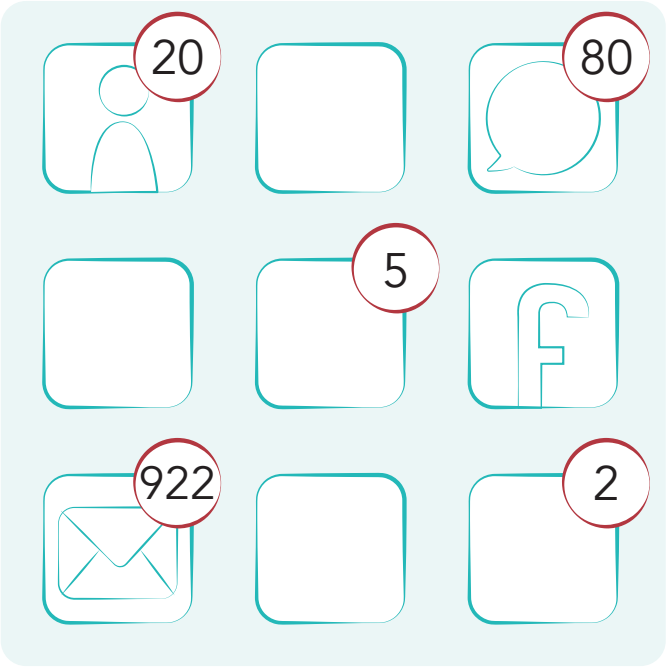

Figure 1 - Overload of notifications

However, both solutions have an additional caveat. Say a person never checks their phone during cooking or does not want to be disturbed while working. Any notifications delivered then will be ineffective. So, the ideal solution is not only to find a moment just before the deadline but to make sure that moment is an acceptable moment to notify.
Preliminary research has shown that there are many applications that attempt to use knowledge about their user's activities. The following examples are of existing products and applications that combine user and device information in order to provide smarter notifications.

> ***Olisto/IFTTT*** *[10], [11]*
> *Can combine date, location and smart device information to, for example, send a notification when leaving home and a specific power consumption is still high (i.e. the TV is still on).*
>
> ***Maps/Waze*** *[12], [13]*
> *Combines real-time traffic information and address in calendar events to provide timely departure reminders.*
>
> ***Timeful*** *[14]*
> *Combines the user's calendar and to-do items to estimate duration of to-do items, plan them in and generate reminders at off-peak times.*

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 [15]–[17], events based on smart devices [4], [18], [19], or a combination of numerous sources of information [20]–[22]. While all very promising, most concepts and implementations predominantly rely on design time logic. Only a few actually manage to create a predictive model that preempts the deadline. Examples of such exceptions, such as Timeful [14], usually create a predictive model and verify this with the user in order to strengthen the model. Nonetheless, not a single of these implementations take the user into account at the time of the notification.

1.1 Problem description

In order to make an application aware of how the user may perceive the incoming notification, knowledge is required about a user's values. Van Riemsdijk introduces the concept of a Socially Adaptive Electronic Partner (SAEP) [23]. A SAEP follows the ideology that technology should adapt to the user and not vice versa. This is achieved by providing methods in which applications can be made aware of a user's values. Through this logic it is possible to gain an understanding about finding an appropriate moment for notifying the user, rather than just focusing on the timing.

The problem of finding an appropriate moment for notification boils down to a number of steps. For starters, in order to provide a reminder notification before a certain deadline, it is imperative that this deadline is known. Consecutively, the preceding activities have to be identified. For this, an existing smart reminder system or predictive model has to be chosen. This, in turn, can only be done once there is sufficient knowledge about the user's activities of daily living (ADL).

Ultimately, the existing model can be extended using user values in order to provide more appropriately timed notifications.

1.1.1 Research scope

Prior to being able to establish the research questions, the scope of the research should be limited since the problem itself is very broad. Most notable, since the area of activity recognition is a rapidly evolving one. However, the current state is that any form of activity recognition based on raw sensor data is still very limited or inaccurate in general solutions [24]. Accuracy can be improved by having location specific setups [22], 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 [25] 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 a substantial enlargement of the scope of research. As such, a choice was made to make use of existing datasets or data streams hat directly provide information about the user's ADL. However, to not limit the applications of the designed concept and implementations, instead the focus was shifted from the aspect of activity recognition and placed onto a proper form of implementation.

1.2 Research question

Following from the concepts discussed previously, the focus of this thesis will be combining the concepts of existing models and trying to extend them with the concept of user values. This leads to the main research question:

> *How can a smart reminder system be extended to incorporate user values to provide more appropriately timed notifications?*

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.

While it was discussed before the numerous existing smart reminder systems exist, it was found that only a few actually use a predictive model to analyze future activities. This, among others, are requirements that need to be decided analyzed in pursuit of an appropriate model.

> *R1: What are the requirements for the smart reminder system?*

Subsequently, existing concepts and implementations should be tested and compared for these requirements.

> *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 and to ultimately improve the timing of the notifications. Of course, for this knowledge about user values is required.

> *R3: How can user values be analyzed and quantified?*

> *R4: How can a smart reminder 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 dispatching a notification. This model can subsequently be implemented in a piece of software in order for the model to dynamically adapt to new input of the user's ADL or values. In order to make the solution more generic, it is important to analyze how the implementation should be structured.

> *R5: How should the smart reminder model be implemented in order to allow easy collaboration with third party software.*

Eventually, the improvement through the inclusion of user values should be tested allowing an answer to the final sub-question:

> *R6: Does the use of the value-extended model provide more appropriately timed notifications?*

1.3 Approach

Below is a sneak-preview of how all aspects of this thesis were approached. After a literature review to analyze existing concepts and implementations, a concept was designed and implemented of which the basic structure is shown in Figure 1.2. Ultimately, the concept was evaluated according to various scenarios to show its added value.

Literature review
In order to answer the first four sub-questions, existing literature was discussed and compared. Ideally, an existing smart reminder system was sought that already takes into consideration the timing of the notification as well as many user and environment variables as possible, while being able to adapt to changes at runtime. In <vul sectienummer in XXX > the reasoning behind these requirements were discussed. Subsequently, in <vul sectienummer in XXX >, the existing papers and implementations were compared. This comparison resulted in the conclusion that no adequate systems exists and therefore concepts would have to be combined.

Leading from this and considering no actual analysis of sensor data is done, several methods of data acquisition were discussed. A choice was made to use a chosen dataset, but keep in mind a possible data stream from a third party. This was motivated in <vul sectienummer in XXX >.

Simultaneously, an approach to incorporating user values was researched and discussed in <vul sectienummer in XXX >. Rather than looking at different types of user values at different moments in the decision process, it was argued that if was sufficient to only look at the annoyance of the notification during a certain activity. Thereby, directly linking activities to values. The subsequent quantization could then be done through an easy questionnaire.

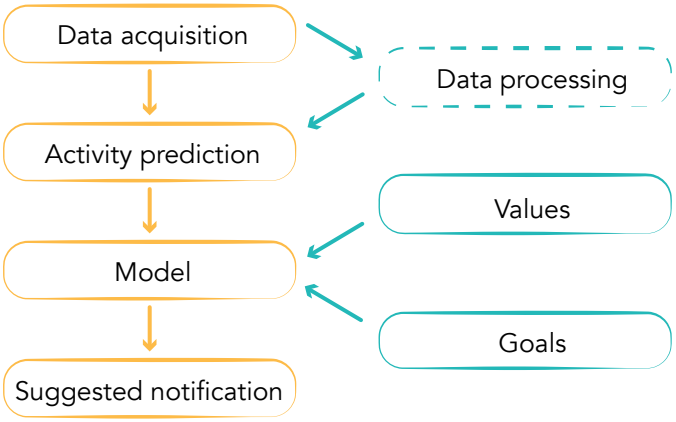

Figure 2 - High level overview of the concept

Concept design and implementation
A concept was designed, based on two main components. First, a one-step algorithm for activity prediction was created, based on a past paper which uses clustering and data mining algorithms. Second, successive predictions were combined with the quantified values in a statistical model based on Markov chains. Using this, predictions could be made regarding possible moments for notifications. These predictions regard two important properties of the moment. Firstly, the expected time between the moment and the deadline, which was desired to be minimized. Secondly, the probability of actually executing the chosen activity corresponding to the moment, and hence the (mathematical) expected value. The latter, which was desired to be maximized. All of the aforementioned considerations were discussed in <vul sectienummer in XXX >.

An implementation was required to do proper testing. However, rather than creating a local test suite, a client-server-based implementation was set up. This allowed for easy data management, client-side input and connections to other data sources, all accessible through an API. The details and reasoning behind all choices were explained in <vul sectienummer in XXX >.

Experimentation
In order to test the designed concept, several test scenarios were established and unleashed upon the dataset. All scenarios were tested in two aspects: success rate (the number of times a notification was successfully dispatched before reaching the deadline), and a score analyzing how well the timing and user values were upheld. The actual results and the details of the scenarios can be found in <vul sectienummer in XXX >.

2. State of the art

This chapter analyzes all aspects necessary to answer the first four research questions before designing the initial concept. First, numerous related concepts, papers and implementations are analyzed and discussed with respect to requirements for a good smart reminder system. In conjunction with this, the concept of user values and their inclusion in a smart reminder system has to be analyzed before any combined concept.

2.1 Requirements of a smart reminder system

There have been various approaches as to how and when to provide feedback to the user. Generally, the preferred method of feedback is the concept of smart reminders [26], a reminder that takes into consideration aspects of the user or their environment. Simultaneously, the timing of the reminder should be sufficiently close to the deadline, the moment before which the reminder should have been dispatched. Newly developed products as well as scientific papers frequently focus on finding novel ways of combining information from smart devices into producing reminders. Examples include combinations of location and time [15]–[17], events based on smart devices [4], [18], [19], or a combination of numerous sources of information to provide insight about a user's activities [20]–[22]. There are various properties that characterize these and other concepts and determine what makes the reminders they provide truly smart.

2.1.1 Notification producing or scheduling

The aforementioned, and many more, concepts may all be loosely categorized in two categories. Firstly, the notification producing concepts. Such concepts react to a number of events (as programmed at design-time) that have happened and subsequently dispatch a notification. Secondly, the notification scheduling concepts, which intercept such notifications and perform a run-time analysis about the deadline and current user context prior to actually dispatching the notification. For example, checking the notification for priority or checking the user's current availability. Furthermore, such concepts generally work post-factum, thereby possibly losing most of the value of the reminder.

In principle, the notification interception analysis can be an extension of the original notification producer. However, these two concepts are generally approached separately. In order to arrive at an appropriate moment for notification, the interception analysis is of most importance as it can be built on top of any notification producing system. Hence, if no sufficing, existing system can be found, this aspect must be built from scratch, but then the chosen notification producer should allow for incorporation of as many user and environment as possible, to allow for a more complete system.

2.1.2 No specific setup

Quite frequently, studies in papers use specific setups to prove a relatively constrained problem. These setups usually comprise of pieces of hardware not usually found in users' homes, even smart-homes, as 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. Considering a more general solution is desired, such concepts should be filtered out.

2.1.3 Values

The idea behind this thesis was to find existing systems and extend them with the concept of values. Therefore, it was interesting to check the incorporation of values, in any form, in any of the analyzed papers and concepts.

2.2 Comparison of existing concepts

Numerous papers, existing products and other concepts were compared to the requirements mentioned above and displayed in Table 2.1. These were all concepts found by querying value-based reminders or smart reminder systems. Short summaries of the important aspects of the concepts may be found in appendix 8.1.

As expected, most the most prominent smart reminder systems made use of only user and environment contexts for design time reminder production. While such systems may be very smart, they all required a rigid set of rules that have to be fixed at design time. For example, with Olisto [10], a rule may be set to get a reminder to take the umbrella when leaving the house, or at a certain time, but never just before leaving. Still, such concepts could come a long way. While not being able to preempt a deadline, it may still be possible to fake this behavior. For example by notifying the user right after they lower their thermostat. This, however, was not a generic nor foolproof solution.

Furthermore, many solutions which analyze specific user behavior required specific setups. For example, MagHive may remind the user to take their keys if noticed the user is leaving. However, it only works with the specific, tagged, set of keys. Similarly, it will not work if the user leaves through a different exit.

The majority of the concepts used information about the user or their environment to some extent. Nonetheless, most of these solutions used this information at design time. There were only a few which took it further and used predictive algorithms or other methods in order to create a system that could adapt at run-time. While most of these concepts provided insufficient information about the algorithms used in order to be used as a basis, there were some that did. Most interesting was CogKnow [20].

Cogknow was the only one which implemented user values, except not in the desired manner. Instead, they are used to identify the required support and hence the necessary reminders. A distinct number of support scenarios are handled and rulesets are defined accordingly. The rulesets are aimed at avoiding interruptions of important activities, but don't do any further analysis.

Furthermore, the Timeful application would have been a good basis for a smart reminder system since it would look for available moments based on the user's calendar and used a self-learning algorithm to improve scheduling of notifications and new activities. Unfortunately, the application as well as its documentation were no longer available and therefore could not be used as a basis.

So, what was useful? There was no existing implementation that could immediately be extended with user values. So in order to arrive at a concept, rather than extending an existing smart-notification system, it had to be designed from scratch. Nevertheless, the ideas from many papers could be incorporated. Furthermore, by designing the implementation in such a way that it was open for third-party applications, a connection with existing systems such as Olisto [10] or IFTTT [11] would allow for easy extension to include many user and environment properties such as weather or heart rate, for example.

Concept	NP	NS	SS	Loc	Act	Tim	Env	UV	Ref.
AHCS	X			X	X	X	X		[22]
CAMP	X								[27]
CybreMinder	X			X	X	X			[28]
Gate reminder	X		X				X		[18]
IFTTT	X			X	X	X	X		[11]
MagHive	X		X				X		[29]
MLCARS	X			X					[30]
Olisto	X			X	X	X	X		[10]
PAIR	X				X				[3]
SRS	X			X	X				[31]
TAFETA	X			X	X	X	X		[5]
CogKnow	X	X	X	X	X	X		X	[20]
Timeful	X	X			X	X			[14]
Attelia		X	X						[2]
Decision maker		X		X	X	X	X		[32]
Fizzy lingustics		X			X				[33]

Legend

NP:	Notification Producing	Act:	User activity
NS:	Notification Scheduling	Tim:	Time
SS:	Specific setup	Env:	Environment
Loc:	User location	UV:	User Values

Table 1 - Comparison of existing concepts

2.3 Preempting the deadline

Since no existing implementation was available, a concept had to be designed from scratch. The core aspect of this concept was that it had to be able to preempt a deadline. This itself brought forth three main problems. First, what is a deadline? Second, how can future activities be predicted. Third, how can a future activity be selected that is relatively close to the deadline.

2.3.1 What is a deadline?

Describing a deadline or a goal can be difficult to describe clearly, especially when ultimately trying to express this in an algorithm. Usually, reaching a deadline comprises of doing a few subsequent activities. That is where deadline reasoning [34] 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 deadlines 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'.

2.3.2 Activity prediction

In order to analyze the possible moments before the deadline, these activities had to be predicted. Prediction algorithms are plentiful, but the field of activity prediction is still relatively new, partially due to activity recognition still being underdeveloped as mentioned in section 1.1.1. There are, however, several papers [35]–[37] which specifically revolve around activity prediction in a support context. Most notably, Nazerfet et al. [37] uses a combination of clustering and association rule mining to attempt accurate prediction temporal relations between activities. Aside from this, all used algorithms, parameters and formulas are clearly described, allowing for easier reconstruction and implementation. This was very useful since activity prediction was not the main focus of the thesis, preventing too much scope creep. However, the concept described by Nazerfet et al. only accounted for single step prediction, hence still requiring a mathematical model to combine the predictions into a predictive model.

2.3.3 The appropriate time – a statistical approach

Before a value based approach was considered, the problem of selecting an appropriate moment already existed. As mentioned in the original problem statement, the ideal scenario would be to receive a reminder shortly before the deadline. However, a simple predictive model would only identify activities with a high chance of occurring. Originally, the hope was that this would have been implemented in an existing smart reminder systems that would have been used as a basis, such as Timeful [14]. Since the system is designed from scratch, the problem still exists.

A solution was found in the use of (absorbing) Markov chains [38], [39], by considering the different activities and temporal relations as states and state changes. The mathematical properties of Markov chains could then be used to calculated the expected time between the deadline and possible moments for notification.

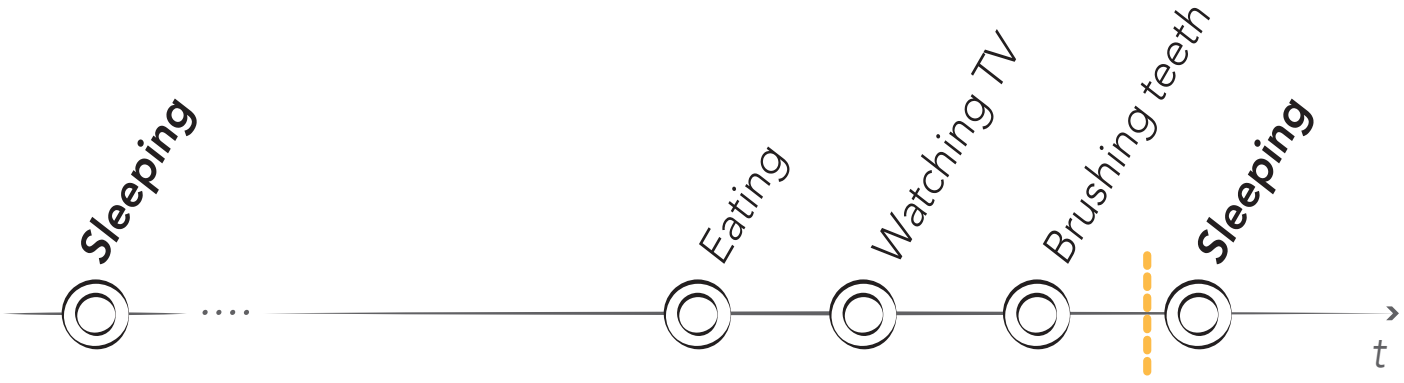

Figure 3 - Time line

2.4 User values

As mentioned in the introduction, the idea of using user values as a method to identify the most appropriate moment for notification stemmed from the concept of a SAEP as presented by Van Riemsdijk [23]. Directly following the work of a SAEP, Tielman [40], Pasotti [41] and Kließ [42] further analyzed the relations between actions and their values.

While their ideas went much further, the idea that certain actions may promote or demote certain actions is crucial. Furthermore, in order to do computational or statistical analysis, they argue that it is possible, although not entirely accurate, to assign quantifiable value gains and losses to specific activities.

2.4.1 Which values

Then for which values should the gains and losses be analyzed? Schwartz [43] proposes several basic human values, but these are very abstract. Govindarajan et al. [44] instead start with 5 simple, core value: peace, truth, love, non-violence and right conduct. Subsequently, other values may be derived from this, such as right conduct leading to hygiene, punctuality, etc. Looking at other papers that implement values such as [40], [45] similar values are used. As such, there is not really a limit to the values used. In any value related concept, a clear pool of values should be selected.

3. Concept design

As no prêt-à-porter solution was available which can be extended to incorporate user values, a concept was designed that built on the fundamentals of other concepts. The initial design was primarily based on combining three concepts. Firstly, a paper by Agrawal et al. [37] which discussed 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. Secondly, Tielman et al. [40] latter paper focused on values and how they link to activities. Lastly, absorbing Markov chains [38], [39] are used to describe the statistical model. This chapter describes all aspects of the concept. However, first a high-level overview is given.

3.1 High-level overview

First and foremost, the data of the user's ADL was gathered and processed where necessary. Through the machine learning concepts of [37], future activities were predicted. Performing these calculations for all possible starting activities would lead to a statistical model. This model could then be extended with knowledge about user values and the ultimate deadline in order to dynamically select an appropriate moment for notification based on the user's activities as they are recorded.

Specifically, the statistical model takes the most recently recorded activity and predicts the probabilities of the user performing a specific other activity before arriving at the deadline. Each activity is then scored based on their corresponding user value and the expected time until the deadline. In this, it tries to maximize user value and minimize the expected time until the deadline. If the highest scoring activity indeed follows, the notification is dispatched. Else, the process is repeated until the deadline is reached.

To understand this concept fully, first the individual concepts are explained and consecutively the combined design is revisited.

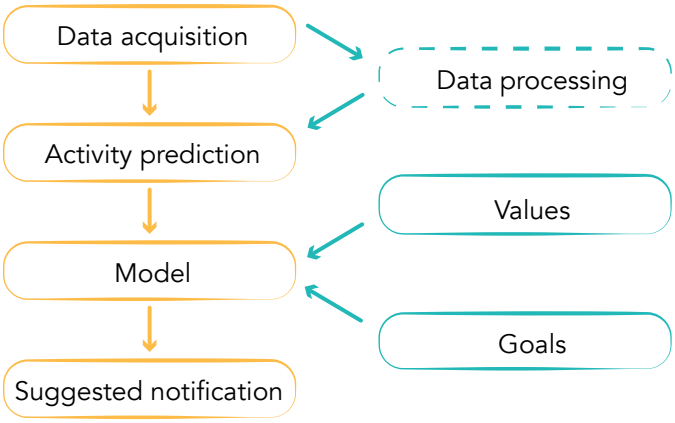

Figure 4 - High level overview of the concept

3.2 Processing incoming data

3.2.1 Data acquisition

As mentioned previously in section 1.1.1, no focus was 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 may be obtained. The first solution would have been to write such a middleware from scratch. This would have been the most labor intense solution. However, if the other data sources are not easily implementable or require extensive processing, using an existing middleware may actually be a faster solution, as well as being more complete. Arcelus et al. [5] did just that; they designed their own middleware. However, it could not be used since it remained exactly that, a design. Hristova et al. [22] instead used an existing middleware: The CASanDRA framework [24], in combination with a context toolkit [46], also used in CybreMinder [28]. The CASanDRA framework, however showed promise due to its broad usage within the research group as shown by many related papers. Nonetheless, 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 could be used. Thanks to close ties with the company behind Olisto [10], access was granted to all services and code behind. Using their information would provide direct insight into events (such as device alarms), states of devices (such as door open or closed) and services (such as weather information). Since Olisto is already an up and running platform, so lots of data is readily available. This also introduced the main downside of using a data stream. Its live data is fairly random and very much depends on the various devices the users own and have connected to the service. While a connection to Olisto would be a very desired next step, pure testing would be easier on a fixed dataset.

Aside from gathering and analyzing data ourselves, the easiest but least extendible option was to use one of the numerous existing datasets scattered over the internet. A select number of these directly provided the desired ADL information. In the Decision Maker concept by Corno et al.

[32], dataset [47] was used, but was synthetically enhanced to add several properties such as the user activity other than call information and mobile phone usage. As such, it was less interesting in its available, original form. Three other datasets had been found and were readily available. These, and similar, datasets could be used both for design and for testing.

3.2.2 Dataset

Three readily available datasets, [48], [49] and [50], 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 datasets 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 dataset, [50] is chosen.

The dataset that was ultimately used is one constructed by Sztyler and Carmona [50]. It was chosen for its well-structured file format, relatively large number of different activities, and the fact that it follows more than one person in more than one situation. In total, more than 6000 data points over four different users have been recorded. The other two datasets scored less on all aspects. Furthermore, the roughly 1500 entries per user correspond to about a month of recorded activities which should be enough to analyze patterns in the daily behavior of the user. Lastly, while considering only four users is not enough to statistically generalize the results. Nonetheless, having just a few different users should suffice in limiting anomalies caused by possible strange habits of a single user.

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." A further description of the dataset can be found in appendix 8.2. Furthermore, a complete list of unique activities can be found in appendix 8.3.

3.3 Activity prediction

Activity prediction was done based on the TEREDA paper by Nazerfet et al. [37]. It focused on two concepts to create a model for activity prediction; clustering and association rule mining. Clustering was done to improve the accuracy of the prediction model and eliminate possible outliers. Association rule learning was used for the actual prediction step, calculating the most likely next activity (cluster). The specific algorithms used are the Expectation Maximization [51] and Apriori [52], [53] algorithms.

3.3.1 Description of the Expectation Maximization algorithm

Duidelijke beschrijving als textbook example??

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. 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,

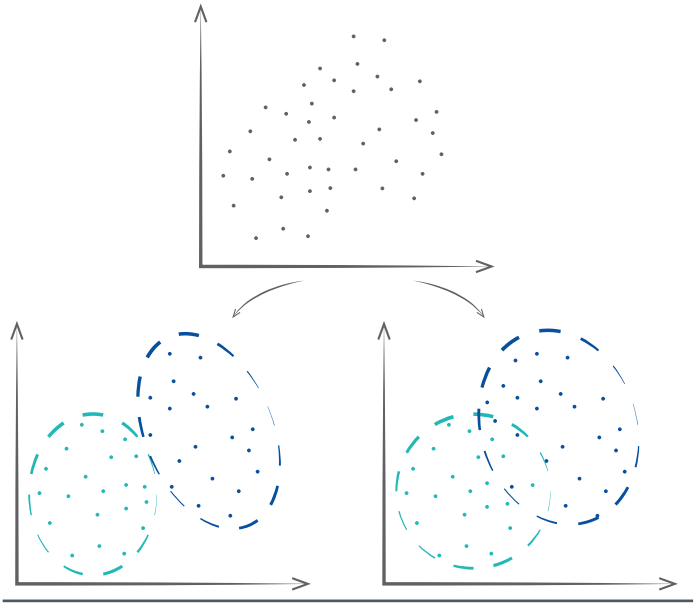

Figure 5 - Difference soft/hard clustering

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. Before being able to use these formulae, the parameters of the models need to be known. This leads to a "chicken and egg" problem. The algorithm solves this problem for any n-dimensional dataset by first performing a random estimate (expectation) to the initial parameters and iteratively improving (maximizing) them.

3.3.2. Clustering of activities

Applying the Expectation Maximization algorithm on activities, Nazerfet et al. [37], performed the clustering of the activities based on their starting time. For example, eating in the morning is considered a different cluster than eating in the evening. Considering how the expected following activities are likely to differ between the two instances, it is clear how this would improve a predictive model.

Additionally, when calculating the cluster parameters, the normal distribution parameters are also calculated for the duration. Activity instances that do not fit within 95% of the area under the curve ($\mu \pm 2\sigma$) are considered outliers. Outliers are not taken into consideration for cluster parameter calculations.

After calculating all clusters, all activities are attributed to their best fitting cluster and fed into the Apriori algorithm

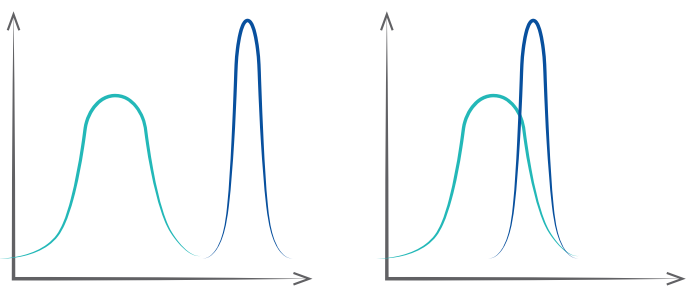

Figure 6 - Overlapping??

3.3.3. Description of the Apriori algorithm

Duidelijk verschil als textbook example

The Apriori algorithm is a machine learning algorithm used to find patterns in large datasets. Specifically, the patterns of frequent item sets. At its core it attempts to identify frequent item sets 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. Nonetheless it may equally be applied to the ADL of a person.
The algorithm is based on the concept of a transaction. A transaction may very broadly defined; it can be a customer purchase at a store consisting of one or more items, or it may be a number of subsequent activities performed by a person. Every transaction, customer purchase, or activity, is logged in a database for further processing.
As a starting point, only individual instances are considered. Examples of instances would be a single bought item, or a single performed activity. A breadth-first search is done to find all such instances that occur in a transaction a minimum number of times; the threshold or support. In the second step, such an instance is considered as a next starting point; for example, the activity sleep. All transactions that contain this instance are then reviewed in a similar manner to find all following instances that occur more frequently than the support. For example, this may result in the set of instances {sleeping ⇒ toilet} appearing frequently in transactions. This process is repeated for the ever-growing sets of instances until no new sets are observed.

Using these frequent instance datasets, association rules can be generated. The association rules can be described using numerous measures. Among others, there are confidence, lift and conviction [54]. Firstly, the confidence of an association rule indicating X leads to Y, or X⇒Y, is the indication of how often the rule has found to be true. The previously defined support, the indication of how often the set of instances appears in the dataset, can be described as:

$$supp(X) = \frac{|\{t \in T; X \subseteq t\}|}{|T|}$$

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

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$$

where X ∪ Y is the union of the instances in the two sets. Rewritten in probabilities, the support can be seen as simply the probability of an event $P(E_X)$, where E_X is a transaction containing item set X. However, since X ∪ Y regards the instances in a set, it can rather be written as $P(E_X \cap E_Y)$. Linking to Bayesian formulae, the confidence can be seen as an estimate of the conditional probability $P(E_Y \mid E_X)$. The drawback of the confidence measure is that it only takes the popularity of set X into account.

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

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)}$$

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 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:

$$conv(X \Rightarrow Y) = \frac{1 - supp(Y)}{1 - conf(X \Rightarrow Y)}$$

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 several measures more. Note that there are more measures of interestingness than just those described above, including, but not limited to, collective strength [55] and leverage [56].

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.

3.3.4 Prediction of future activities

Applying the Apriori algorithm on activities, Nazerfet et al. [37], followed the algorithm as described before. However, they imposed two limitations. First, their implementation of the algorithm focused solely on the support and confidence for identifying sets of instances. Second, only pairs of subsequent activities were considered with no deeper analysis being performed.

The reasons for these limitations were twofold. Firstly, the concept of transactions of store-bought items cannot be applied perfectly on the concept of subsequent activities. While the purchased items are completely independent, the activities are linked by time. However, the similarities are substantial enough to warrant using the Apriori algorithm for the concept of activity prediction. Secondly, since Nazerfet et al. were only interested in the directly successive activity, further expansion of the instance sets was not necessary. These two reasons do not fully explain the imposed limitations. Still, Nazerfet et al. did not provide further clarification behind their reasoning.

Nevertheless, following the positive results of the experiments performed by Nazerfet et al., the choice was made to follow their choices for both clustering and the prediction as a starting point for the model. This would avoid scope creep due to perfecting the prediction algorithm as well as allow future readers to easily reproduce the results as mentioned further on in this report.

3.4 Value based design

As explained before, humans make many decisions based on their norms and values. In order for a system to be able to mimic such decisions, the technology needs to have a notion of values as described by Van Riemsdijk [23].

Thanks to their generalizability and stability over time, values are perfectly suitable for identifying underlying reasons for actions [57]. Formalizing this relationship is complex and can be done in a number of different ways. The simple way used in this report followed that of Tielman [40] and Pasotti [45] 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 was made that for different actions, the values are commeasurable. This assumption "is not a trivial one", but is frequently used to allow quantitative comparison between otherwise incomparable numbers.

The logical step would have been directly assign values to all activities in the dataset for further calculation. However, this is not necessarily useful to the cause. Instead, it is important to revisit the goal of the thesis before deciding upon the implementation of values.

3.4.1 The appropriate time – a value based approach

The goal of the thesis is to find the most appropriate time for notification. There are two aspects to this problem.

Time aspect

As mentioned before in 2.3.3, in the ideal situation, the notification is dispatched just before the deadline. The reasoning behind this can be explained in terms of values. The value gain achieved by the reminder should only count in the calculation when the user actually remembers. Since this is difficult to test and quantify, an assumption had to be made. It was assumed that a longer time between the notification and the deadline would have a negative effect on the value gain caused by the reminder since the reminder would have a much lower chance of causing the user to remember.

Another option would have been to introduce a cut-off. In this case a reminder sent too long before the deadline would be considered a failure. The downside of this is that no distinction is made between a reminder sent just after the cut-off moment, or just before the deadline. Furthermore, scoring the result based on time would allow for optimization with regards to this variable.

Value gains and losses

The second aspect of the problem was finding a moment that results in the largest value gain, as may be concluded from the previous section. For this, the concept of quantifying the value gains and losses as discussed by Tielman [40] is accepted. There are two aspects which comprise the ultimate value gain:

The value gains invoked by remembering
vs.
The value losses invoked by the annoyance of the notification

When considering a scenario, the absolute value gain invoked by remembering is not of importance. Why? Because throughout all analysis it remains a constant. It may only change if the user does not actually remember. However, this was already covered in the time aspect. Therefore, the problem summarized to analyzing the value losses invoked by the annoyance of the notification.

3.4.2 One type of value

Analyzing value gains and losses required two things: a set of values to analyze and a method to obtain quantitative data regarding the values. As mentioned in 2.4.1, even the decision of which values to analyze is a difficult problem. The following example scenario was considered:

> *Assume values are quantified from -5 to +5. Peter has set a reminder to close his garden doors before going to sleep. Normally he would be likely to forget. However, if he remembers, he feels much safer. Hence, his value of safety is strongly promoted (+3). The notification may be sent out either while watching tv (harmony +1, excitement +1) or while he is brushing his teeth (health + 2).*

If Peter would have received a notification during either of these activities, one of several things may happen:

- He may quit this activity, thereby nullifying any value gains
- He may be disrupted by the notification, thereby causing a reduction in the value gains
- He may be disrupted by the notification, thereby introducing value losses (e.g. harmony)
- He may shortly acknowledge the reminder but not mind it.

The difficulty in this lay in how to compare the different value gains and losses. As mentioned at the start of this section (3.4), assuming commeasurable values may have posed a solution. However, the question was whether such a complex approach is necessary at all.

The values losses as invoked by the nuisance of the notification were directly related to the gains invoked by the activity. Rather than seeing them as separate, the four scenarios mentioned above were combined by saying Δv is analyzed, the change in values. Psychologically losses weigh much heavier than gains [58]. As such, Δv is predominantly determined by the losses invoked by the nuisance of the notification. However, establishing these values for all possible activities would have been a very difficult task by itself, especially since they differ per user.

Instead, rather than analyzing the losses invoked by the nuisance of the notification, an intermediary may be removed. The result would then be to look directly at the nuisance of the notification as a single value. Other ways of describing this could be emotional well-being, or harmony. It would be much more feasible to ask a user about how annoying a notification is during a certain activity, rather than all individual values. Having focused on this single value, may well have be sufficient.

The big advantages of looking at just a single value come down to three aspects:

- No comparisons between different values had to be made, removing the need for the assumption of commeasurability.
- It would be easy for a user to supply quantitative data regarding values of annoyance because they are more concrete than most other values.
- Furthermore, statistical analysis is much simpler using only a single value variable.

3.4.3 Quantifying values

Since linguistically it was difficult to distinguish between a type of value and a quantified value, from here on the following definitions were introduced:

> ***Quantified user value (QUV):***
> *The quantification corresponding to the given user value; i.e. the value of the value.*

To perform calculations based on the QUVs, they first had to be obtained. The ideal method would have been to start with an educated guess and increase or decrease the QUV based on user feedback. Nevertheless, this would still require research or previously entered QUVs. For now, this self-improving aspect was ignored to prevent possible noise in results and keep a straight-forward implementation.

Instead, the user is presented with a questionnaire which directly assigns scores to activities. As previously it was decided that only the nuisance of the notification would be analyzed, only a single choice per activity would have to be made. For every activity, a choice is made among five options to answer how annoying it would be to be notified during this activity, corresponding to common questionnaires and a Likert-type scale [59]. The options are showing in figure xxx.

Activity	Unacceptable	Very annoying	Annoying	Somewhat annoying	Not at all annoying
toilet	○	○	○	○	●
sleep	●	○	○	○	○
groom	○	●	○	○	○

Figure 7 - Possible score choices

Ideally, any reminding would take place at an activity where a notification is not at all annoying. However, most importantly, no notifications should be dispatched when it is unacceptable. To convert these answers into QUVs, two decisions had to be made considering the range of the QUVs.

First, the zero-point of the range. There were two options for the zero-point. Either 'not at all annoying' would be considered the zero-point, with the other QUVs going into the negative. The other option would be to consider 'unacceptable' as the zero-point, with other QUVs going into the positive. Considering only notification during an activity marked as 'unacceptable' was considered a failure, setting this as the zero-point would allow for later calculations to still consider 0 as a failure while positive QUVs would be successes.

Second, the distance between the various answers. This distance between the QUVs is very subjective. However, a common practice is to set them equidistant. The Linkert-scale suggests simple integer values. As such, assuming 'unacceptable' equates to a QUV of 0, 'not at all annoying' would equate to a QUV of 4, the rest distributed linearly. The assumption that these five answers are linearly distributes is not a trivial one, but warrants further investigation in its own right. Furthermore, weighting may be introduced at a later time, allowing adaption to the user's preferences.

3.5 Statistical model creation – *Markov chains*

Recapping on the status, using the clustering and prediction techniques, given any current activity the next activity could be predicted with a certain probability. To find an appropriate moment, rather than just one step into the future, all possible steps until the deadline had to be analyzed.

As mentioned before, it was accepted that reaching the deadline was synonymous to reaching a certain activity. While it was possible to let the statistical model dynamically adapt to having a single or more of such deadline activities, for simplicity, only a single goal activity was accepted.

The requirements for an appropriate moment were previously discussed in 2.3.3 and 3.4.1. Ideally, a moment should be found that optimizes the following aspects:

- It should take place between now and the deadline (probability)
- The time between the moment and the deadline should be minimal
- The QUV corresponding to the moment's activity should be maximal

Before continuing, another choice had to be made. When analyzing the time between the moment and the deadline, ideally an actual time in minutes would be preferred. With the previous choices made for the clustering and prediction techniques, combining subsequent predictions would account for very inaccurate predictions for durations. If the clusters had been calculated based on durations, this inaccuracy might have been reduced. Still, the combined predictions would exponentially increase this inaccuracy. Therefore, rather than looking at the absolute time, the number of intermediate activities was considered as a replacement for the time. While not an ideal solution, it may be argued that assuming this synonymy is appropriate. Anytime a user switches activity, their focus is shifted and requiring more attention to regain proper focus. This is one of the reasons why multitasking does not really work [60]. While same time may be occupied by a person watching TV as them showering, then going to the toilet and several more short-lasting activities, they do not necessarily have the same effect on the brain. The addition of time in the smart reminder concept was introduced to simulate the chance of the notification having its intended effect of reminding. As such, the choice was made to look at the number of intermediate activities between the moment and the deadline, rather than the absolute time between them.

Combining the above matters, the problem may be stated as:

> *Given a current activity cluster A and a goal activity Z, we are looking for an activity cluster S with the highest QUV, that is likely to occur before the deadline, and that will be reached with only a minimal number of expected steps remaining before Z is reached. So:*
>
> $$A \rightarrow [n\ steps] \rightarrow S \rightarrow [m\ steps] \rightarrow Z$$
>
> *Where the aim is to find a minimal m with a maximal probability and value for S.*

Note that Z is not described as an activity cluster since it is irrelevant which cluster of Z is reached. As such, all cluster for Z may be combined into one.

Procedurally, the simplest way would have been to generate a probability tree, analyzing each activity and its possible successors one by one. However, this would have been a very computationally intensive process. Given a dataset of substantial size, ergodicity may be assumed. Since, given that there are enough recorded activities, it may be assumed that at one point the user reaches a similar activity to one performed before. Such an irreducible system may be modelled as a (discrete time) Markov chain where every activity cluster is represented by a state. The main advantage of using Markov chains is that there are numerous ways to analyze such chains. These ways are computationally much less intensive then analyzing a probability tree due to their reliance on formulae rather than on iterative algorithms. All three aspects up for optimization – probability, QUV and number of steps between the moment and deadline – were calculatable using Markov chains. However, in order to perform these calculations, the problem had to be formally defined, mathematically that is.

Explanation - Markov chain

> *"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." [38], [61]*

3.5.1 Expected value

As mentioned in the previous section, an activity with a high QUV was sought while being likely to take place before the deadline. Essentially, this is mathematically synonymous to saying the expected value (or expected QUV) should be maximized. In the following paragraphs, the basic method of obtaining expected values from a Markov chain are explained.

Mathematically, the expected value of any random variable X is defined as the probability-weighted average of all possible values X can take on [61]:

$$E[X] = \sum_{x:\, p(x)>0} x\, p(x)$$

Given that any activity corresponds to only one QUV, this equates to:

$$E[X] = x\, P(X)$$

where x is the QUV corresponding to activity cluster X. Considering the QUVs were directly taken from the user, all that remained was to find the probability of each state. Knowing all transient probabilities from the prediction algorithm, a transition matrix of the system could be built. As an example, we will take a system of three states that represent three activity clusters as shown in Figure 8.

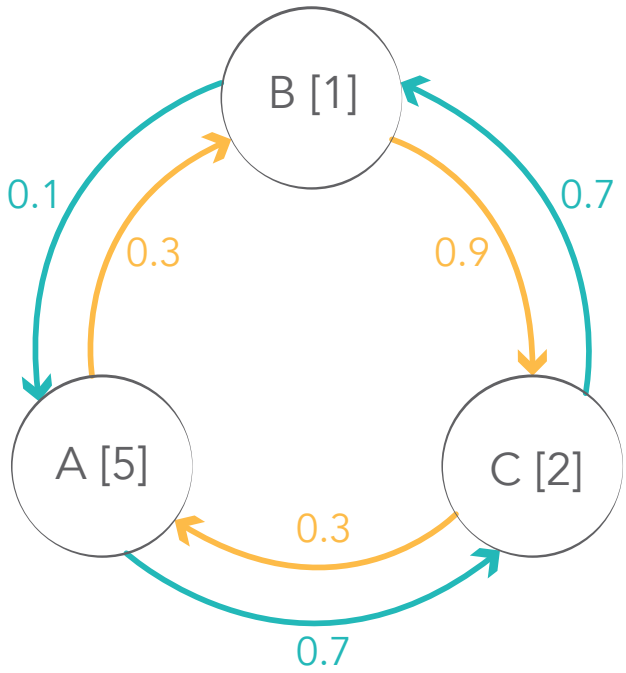

Figure 8 - Example state model with three values

In the model, each state has a certain QEV and a probability to reach a different state next. These probabilities and QEVs could be combined in a so-called transition matrix and a value matrix. For the example model that would result in respectively:

$$P = \begin{bmatrix} 0 & 0.3 & 0.7 \\ 0.1 & 0 & 0.9 \\ 0.3 & 0.7 & 0 \end{bmatrix}, \qquad V = [5 \quad 1 \quad 2]$$

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

As mentioned before, an important characteristic of our system is its irreducibility. In simple words, at any moment it is known 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 to be solved is:

$$\pi P = \pi$$

Where π 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:

$$\pi(P - I) = 0$$

$$(P^T - I)\pi^T = 0$$

This can be solved by finding the eigenvalues and corresponding eigenvectors of the matrix (P^T-I). This corresponds to solving the following equation:

$$\det(P^T - \lambda I) = 0$$

$$\det \begin{pmatrix} -\lambda & 0.3 & 0.7 \\ 0.1 & -\lambda & 0.9 \\ 0.3 & 0.7 & -\lambda \end{pmatrix} = 0$$

Of course, the result will be multiple eigenvectors. The one corresponding to the stationary distribution

is the one for which all entries of the eigenvector are positive. In the example, this would be the eigenvector corresponding to the eigenvalue λ=1, which is (0.38,0.81,1). Consequently, the stationary distribution is:

$$S = \frac{1}{0.38 + 0.81 + 1} \cdot (0.38 \quad 0.81 \quad 1) = (0.17 \quad 0.37 \quad 0.46)$$

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

$$E[\pi] = V \circ S$$
$$= (0.85 \quad 0.37 \quad 0.91)$$

As shown, this is a simple solution to finding probabilities and expected values for an irreducible Markov chain. There is, however, a problem approaching the problem in this manner. Reconsider the problem as stated at the start of this section: Starting at state A, a state S is required to be reached before reaching deadline state Z. However, the model that was just discussed doesn't consider state Z as a final state but simply allows for continuation. Therefore, the acquired probabilities and corresponding expected values do not represent reality. In order to solve this problem, the transition matrix had to undergo a transformation and make the final state absorbing.

3.5.2 Absorbing Markov chain

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 [39]. 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 more difficult to work with, but it can be approached in a similar way as a converging series. Through some simple transformations and calculations, it is possible to calculate the expected number of steps between any state and an absorbing state as well as other interesting properties.

To start, the system must be made absorbing. Continuing with the same example system as before and taking state C as the final state, the new, absorbing, transition matrix becomes:

$$P = \begin{bmatrix} 0 & 0.3 & 0.7 \\ 0.1 & 0 & 0.9 \\ 0 & 0 & 1 \end{bmatrix}$$

It is nothing other than making the corresponding element equal one while all other elements in the row are reduced to zero. The next step is to obtain the fundamental matrix. 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:

$$P = \begin{bmatrix} Q & R \\ \mathbf{0} & I \end{bmatrix}$$

Q: A *t-by-t* matrix, describing probabilities from a transient state to another
R: A nonzero *t-by-r* matrix describing probabilities from a transient state to an absorbing states
0: A *r-by-t* matrix of zeros
I: A *r-by-r* identity matrix

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

$$Q = \begin{bmatrix} 0 & 0.3 \\ 0.1 & 0 \end{bmatrix}, \qquad R = \begin{bmatrix} 0.7 \\ 0.9 \end{bmatrix}$$

The fundamental matrix is a matrix that describes the expected number of visits of a transient state before being absorbed. The calculation of the fundamental matrix starts with the property matrix Q. Entry (i,j) of Q describes the probability of going from state i to state j in exactly one step. The same entry in Q^2 describes the probability in exactly two steps, etc. As mentioned before in 3.5.1, the expected value of anything is calculated by summing all probabilities multiplied by their values. Therefore, the expected numbers of visits, the fundamental matrix, are calculated using equation 3.5.

$$N = \sum_{k=0}^{\infty} Q^k = (I - Q)^{-1}$$

Calculating that for the example, we find:

$$N = \begin{bmatrix} 1 & -0.3 \\ -0.1 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1.03 & 0.31 \\ 0.10 & 1.03 \end{bmatrix}$$

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

$$T = N\mathbf{1}$$

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

$$T = \begin{bmatrix} 1.03 & 0.31 \\ 0.10 & 1.03 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.34 \\ 1.13 \end{bmatrix}$$

Therefore, it can be concluded that it will take an average of 1.34 or 1.13 steps before being absorbed if started in state A or B respectively.

As explained before, the non-absorbing version of the Markov chain did not represent reality in its probabilities of reaching the transient states before being absorbed. However, these probabilities can be recalculated for absorbing Markov chains. This is done using the following formula

$$H = (N - I)N_{dg}^{-1};$$

Where N_dg 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 the example, the following results are obtained:

$$H = \left(\begin{bmatrix} 1.03 & 0.31 \\ 0.10 & 1.03 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \begin{bmatrix} 1.03 & 0 \\ 0 & 1.03 \end{bmatrix}^{-1} = \begin{bmatrix} 0.03 & 0.30 \\ 0.1 & 0.03 \end{bmatrix}$$

From any probability can be found. For example, the probability of passing through B before reaching C, given that the current state is A is described by:

$$P[B|X_0 = A] = H_{01} = 0.30$$

Consecutively, to obtain the expected values, every element in each row is multiplied by their corresponding QUVs:

$$E[S] = [5 \quad 1] \circ \begin{bmatrix} 0.03 & 0.30 \\ 0.1 & 0.03 \end{bmatrix} = \begin{bmatrix} 0.15 & 0.30 \\ 0.5 & 0.03 \end{bmatrix}$$

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

$$A \rightarrow [n\ steps] \rightarrow S \rightarrow [m\ steps] \rightarrow Z$$

The expected value for S is maximized and the minimum number of steps for m is minimized. In order to find the expected value for S, equation (3.15) 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 (3.14). Looking at the example for the last time, taking A as our current state and C as our final state, for intermediate state B the results are:

$$E[B] = 0.30, \qquad T[B \rightarrow C] = 1.13$$

For both T and H, the results are 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 the expected value for S is to be maximized while keeping the number of steps until absorption at a minimum, these calculations must be done for all possible S.

3.5.3 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, there are usually a series of sequential activities. For now, this is not taken into account. A possible solution would be to look at every possible set resulting from the Apriori algorithm and map it as a separate state. Even this is not ideal but it would be an improvement.

3.6 The appropriate time – *the combined approach*

Now that a statistical modal could be constructed, the appropriate moment had to be selected. This was done using the procedure as shown in figure xxx. Once a reminder is requested, the deadline is known as entered by the user. Subsequently, all clusters as well as all predictions from one activity cluster to their likely successor are calculated. Then, based on the user's entered QUVs and the calculated probabilities, the statistical model is calculated. Given the most recently recorded activity as the starting state, the expected value and expected number of steps until the deadline are calculated for all other activity clusters. These two numbers are then optimized according to a scoring function[1]. The highest scoring activity cluster is seen as the most appropriate moment which is also likely to be occurring before the deadline. If the next recorded activity corresponds to this high scorer, the notification should be dispatched. If not, the statistical model and starting state are updated to the newly recorded activity and the process is repeated. This continues until either a moment has been found or the deadline is reached. If the deadline is reached, the notification should be dispatched immediately, but this is considered a failure.

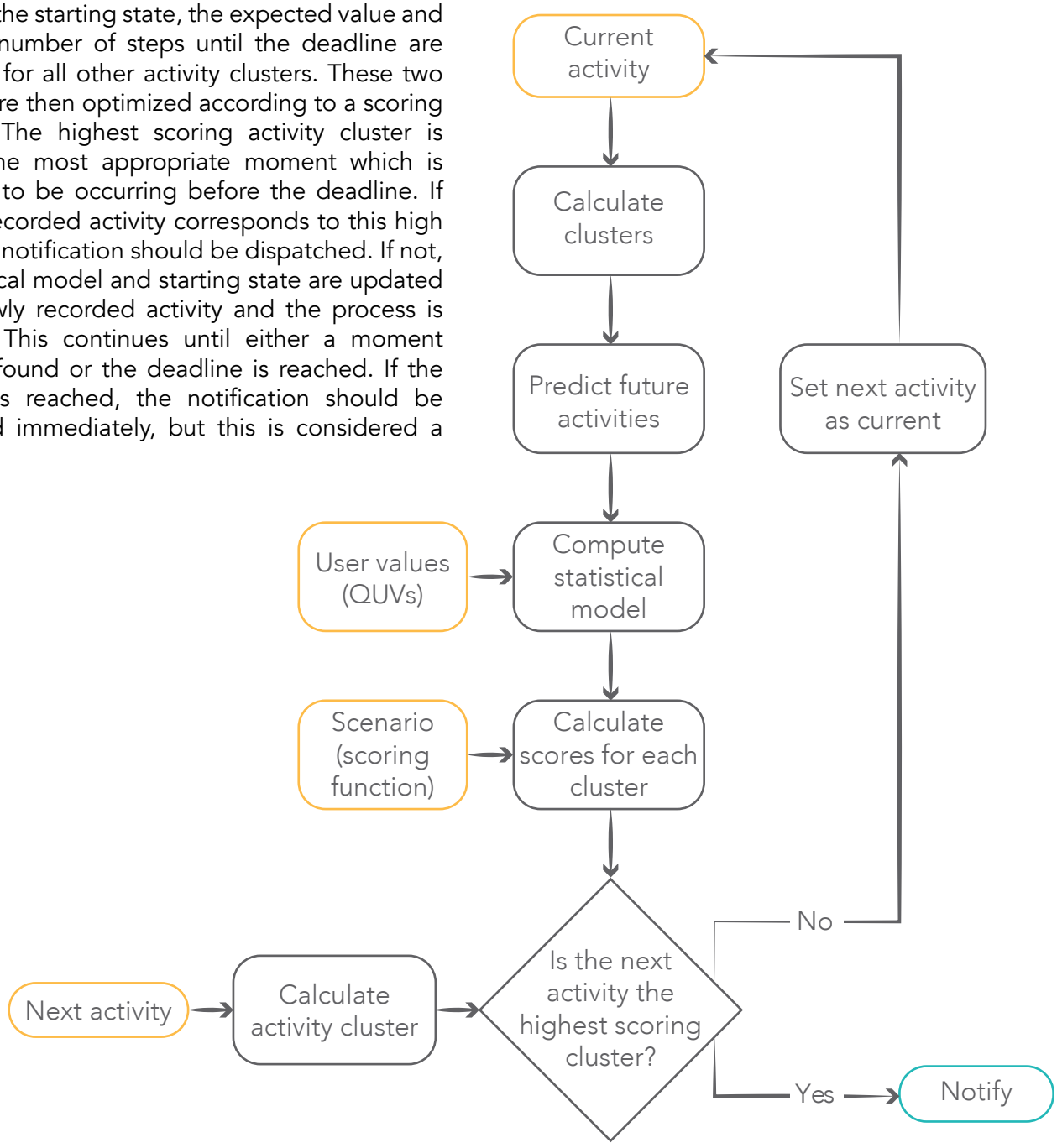

Figure 9 - Decision model

1. This scoring is relatively arbitrary and user subjective. However, it allows for a single measure of comparison. Furthermore, changing the scoring function allows easy adjustment of the weighting between the QUV and the time before the deadline, per the user's preference. The scoring function is further described in chapter 5.

3.7 Concept description

Now that all aspects of the concept have been discussed, the combined design is revisited. 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. The data is normalized to contain type, starting time and duration. For now, a dataset was used, but the possibility of a data stream was accounted for.

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.

Deadline
The goal activity, before which the reminder should be dispatched to the user, should be provided. For now, this was done manually and only a single such activity was selected.

Model
All cluster predictions are combined and modeled as a Markov chain with corresponding transition matrix. The model is made absorbing, based on the provided deadline. 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 QUV of the corresponding activity, as supplied by the user
- The expected number of steps until absorption is calculated

Moment selection
Based on the most recently recorded activity, the deadline and the model, an appropriate moment is predicted. This prediction is done through a scoring function aimed at maximizing the expected value and minimizing the expected number of steps between the moment and the deadline.

While the prediction does not correspond to newly recorded activities, the process is repeated and the prediction is updated. This continues until either the predicted moment occurs, or the deadline is reached. At this point, the notification should be dispatched.

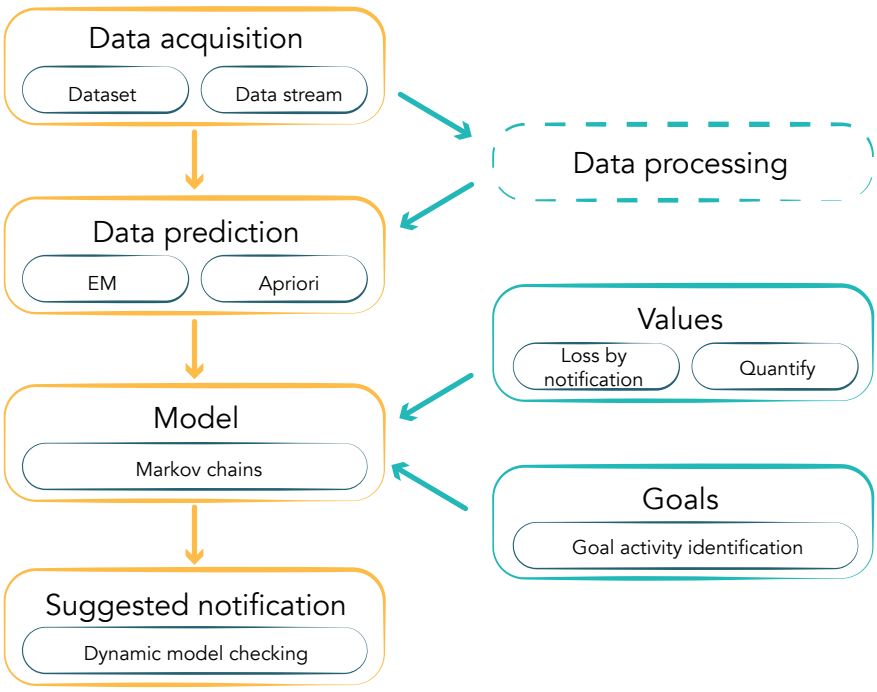

Figure 10 - Detailed concept overview

```
 * @private
 */
function findTransition(lastActivity, clusterModel, predictionModels, cl
    let nextIdx = 0;
    const clusterIdxDict = {};
    const transitionMatrix = Matrix.zeros(clusterCount, clusterCount);

    function getIndex(key) {
        if (!(key in clusterIdxDict)) {
            clusterIdxDict[key] = nextIdx++;
        }
        return clusterIdxDict[key];
    }

    predictionModels.forEach(model => {
        model.nextClusters.forEach((cluster, index) => {
            const singleProb = 1 / model.counts[index];
            const fromKey = model.activity + '_' + index;
            const fromIdx = getIndex(fromKey);
            Object.keys(cluster).forEach(toKey => {
                const toIdx = getIndex(toKey);
                transitionMatrix.set(fromIdx, toIdx, cluster[toKey]*singl
            });
        });
    });

    return transitionMatrix;
}

function defaultScoring(probability, value, steps) {
    return probability * value / steps;
}


**
 * @param {activity}          lastActivity
 * @param {Array.<ClusterModel>}  clusterModels
 * @param {Array.<predictionModel>}  predictionModels
 * @param {Number}            clusterCount
 * @param {ClusterModel}      finalModel
 * @param {Object.<string, number>}  values
 * @param {Function}          scoringFunction
 *
 * @return                    {[startKey: string, scores: Array]}
 */
function findMoment(lastActivity, clusterModels, predictionModels, clusterC
    const clusterModelObject = {};
```

4. Implementation

The implementation was a major aspect of the project. It was used to show the feasibility and attainability of the proposed concept. Due to the field of technology in which this researched is based, it was important to allow for connection to a real-life application for possible future field testing or implementation. As such, several things had to be done.

First, a suitable platform had to be chosen. This platform should not only allow for all desired datasets to be supported, but preferably also allow for a standardized connection to third-party platforms. Secondly, the algorithms of the conceptual design had be implemented in code and therefore, a programming language with the necessary capabilities and libraries was required. Lastly, the implementation had to provide a way for a user to input data (or to upload bulk sets of testing data) as well as obtain results, preferably all in the form of a graphical user interface for easy access.

Subsequently, the actual implementation was done. This chapter will include explanations of the most important aspects of the implementation and its algorithms.

4.1 The actual implementation – *Notival*

The actual implementation of the designed concept was done in the form of an application called NotiVal (VALue based NOTIfications). It serves both a backend for calculations and communications as well as a front end for user input, bulk data uploading, and testing as shown in Figure 11.

The architecture will be further explained in the next sections. While code fragments are not included to uphold clarity of explanation, the full source code can be found for analysis at:
https://github.com/RomanovX/Thesis.
Combining all the aforementioned aspects, a JavaScript based, Node.js server is established, along with a MongoDB database. It serves a frontend used to view statistics and allows for user input. All communication is done through a RESTful API. In order to do complex machine learning calculations and matrix calculations, several library functions are imported.

4.1.1 Architecture

Following the common practices of web-based services, a basic client-server setup was established along with a database connection as visualized in Figure 12. The application runs on a JavaScript based, Node.js server along with a MongoDB database. Furthermore, it serves a frontend used to view statistics and allow for user input. All communication with the user and third parties was done through an HTTP RESTful API. Lastly, in order to do complex machine learning calculations and matrix calculations, several library functions are imported. Such library functions range from simple matrix implementations to the full Apriori algorithm.

While to some, the choices for such a platform may be obvious or trivial, there is clear reasoning behind the choices for this kind of platform and architecture. The unacquainted reader may choose to read appendix 8.4.3 for a detailed description of the technical components of the implementation, as well as reasoning behind all choices.

Going a level deeper, consider Figure 13 which depicts how the most important aspects of the implementation were organized. A clear separation of concerns was done by splitting the frontend, the resource handling, and the actual methods. Most obviously, the most important aspect of the implementation is the moment selection. For this, prediction models had to be created, using the clustering and prediction algorithms. These calculations were done based on the activity and value data as supplied by the user and/or other 3rd party implementations. In order to use existing activity data, this data had to be properly recorded. Given the used dataset, several hiccups arose. These aspects of the implementation as well as the encountered difficulties are explained in the next sections. However, for this it is important to understand the different kinds of resources used in the implementation and stored in the database.

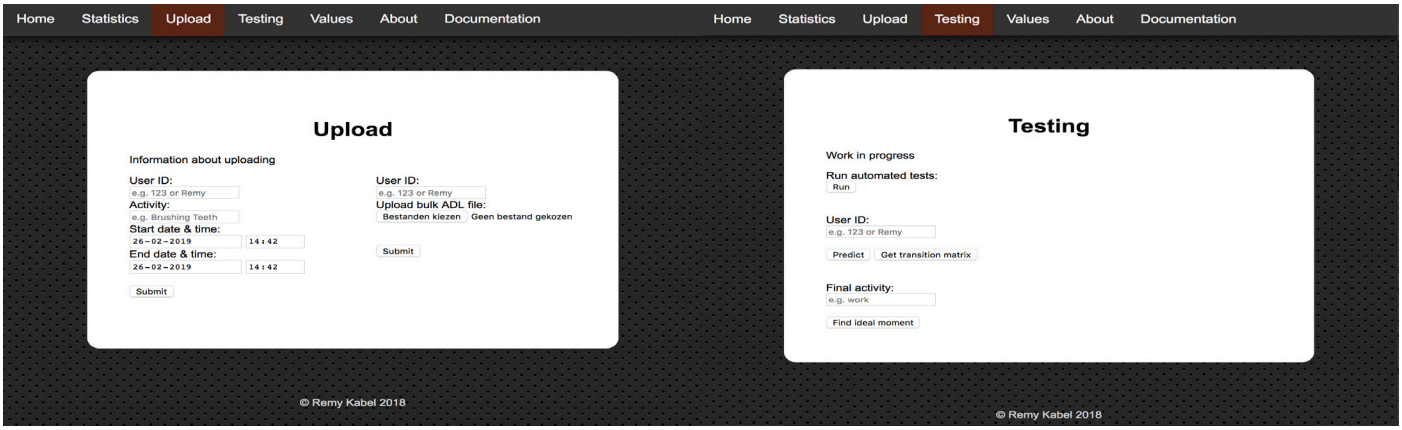

Figure 11 - : Example views of the Notival Application

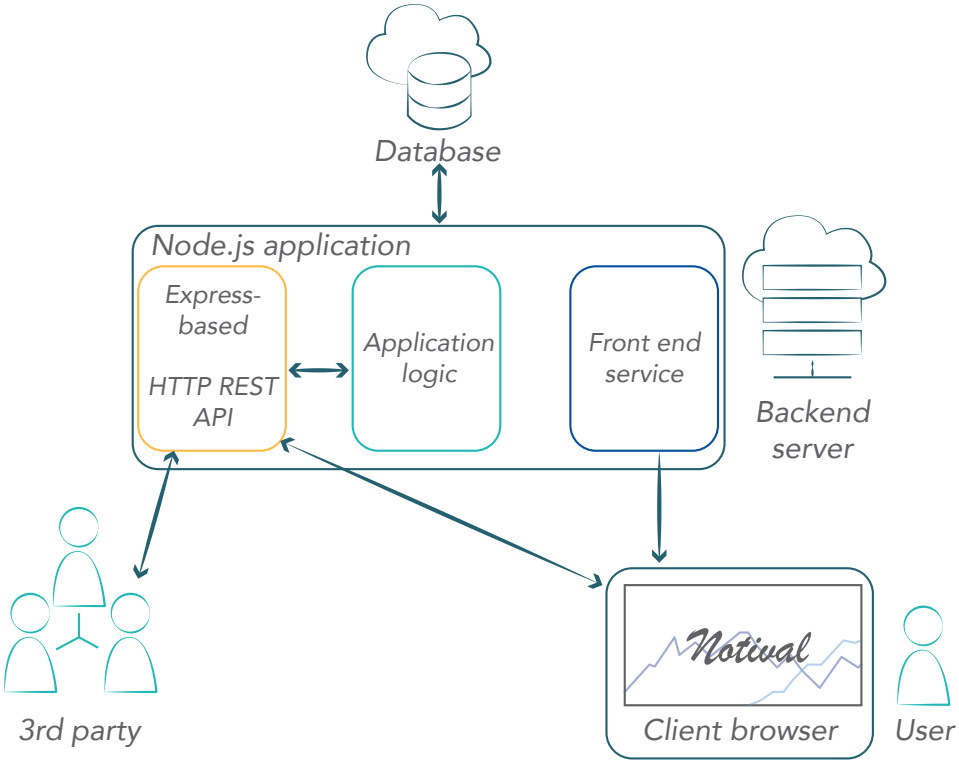

Figure 12 - High level architecture overview

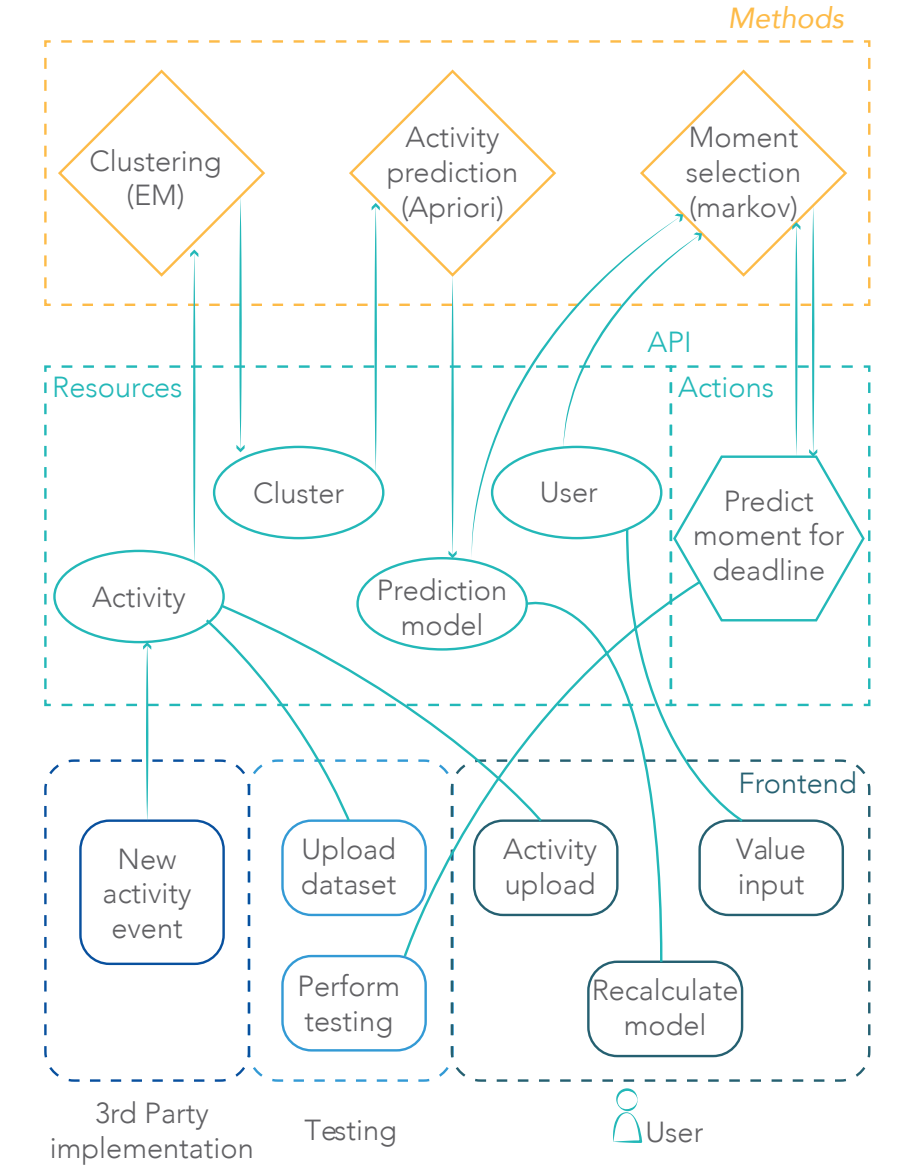

Figure 13 - Low level architecture overview

4.1.2 Resources

Resources are used within NotiVal to contain the most important pieces of data. They are provided and accessible through the API (as shown in Figure 4.2), following the REST principles. Consequently, they are stored in a database corresponding to clearly defined models, allowing for easy use further down the road, preventing errors. These main resources are:

Users

Described by their id as well as their values

Activities

Entries of activities, described by which user they belong to, their name and information about their starting and ending time.

Clusters

The models of cluster in which the activities have been sorted according to the clustering algorithm. They are described by which user they belong to, which activity they correspond to, and the parameters of the model.

Prediction model

Information about the probabilities of each subsequent cluster, given the current cluster as calculated through the Apriori algorithm. Once again, they are further described by which user they belong to.

4.1.3 Analysis of incoming data

Incoming data is analyzed in two moments. Either when a new user activity is recorded through an API call, or when a dataset file is submitted. While the analysis of the incoming data in itself is not very exciting, there are two important aspects to discuss.

Errors & Overlapping activities

As in any possible data source, error may occur. As shown in more detail in appendix 8.2, every recorded activity consisted of a start and end event. Occasionally, one of two things could happen.

First, it could occur that an activity that was started was never completed. This anomaly only occurred twice in the used dataset, so these entries were simply omitted. Second, a new activity could be started before a prior one was completed. This was solved by finding corresponding events rather than assuming that two subsequent events always corresponded.

Analyzing starting times

The third major aspect of the implementation is its clustering. In order to cluster by starting times, these times had to be converted from a date into a number corresponding to the time on the day. The simplest solution was to use the number of minutes since midnight. This brought along one drawback. The time of 00:00 would correspond to 0 minutes. However, a time of 23:59 would correspond to 1439 minutes. While these times are only a minute apart, to the clustering algorithm there could be no bigger difference between these times. As no solution was found for this within the clustering algorithm, it was accepted that a certain degree of inaccuracy would be found in the clustering.

4.1.4 Prediction models

Prediction models are basically the rows of the transition matrix. However, they are more detailed so they could more easily be updated when new activities were recorded. The prediction models were the result of performing the clustering and prediction algorithms. For a large part, library functions were used to implement these algorithms [71], [76]. A similar procedure and similar parameters were used as described in the paper by Nazerfard et al. [37] to acquire these clusters, and later the prediction models.

4.1.5 The appropriate time – *implementation*

To predict the appropriate time, three things were required from the user; a sufficiently large set of recorded activities, a set of recorded QUVs, and a deadline. Subsequently, the actual moment selection was, of course, the most important part of the implementation. Given that the recorded activities were already processed into prediction models, there only one major part remaining in the implementation. This last part was the matrix calculations corresponding to those explained in section 3.5 about the statistical model.

After the code shown there, the scores were calculated based on the defined scoring function and passed back to the system for possible notification. Another very interesting aspect of this section is how adept JavaScript is in the implementation of Matrices, albeit with the use of a library.

4.1.6 Testing

The final part of the implementation consisted of creating a testing suite. This testing suite would allow for randomization of selecting test cases, QUVs, and deadlines. This will, however, be explained in the next chapter.

4.1.7 Conclusion

A separation of concerns was done by splitting the implementation into frontend, resource handling through the API, and the actual methods used for calculations and predictions. Several 'clean-ups' had to be performed to analyze incoming data, and transform it into a general data structure for activities. Activities could then be clustered. From the clusters, a prediction model was made. Combining the prediction models and the user values, the moment calculations could be performed

5. Experimentation

Having established and implemented a model, the final step is to answer the last research question:

Does the use of the value-extended model provide more appropriately timed notifications?

While having a proper implementation shows that the concept is achievable, it is more interesting to see if the model shows improved performance. This chapter covers the approach and method of the testing and shows the results of the tests performed.

5.1 Introduction

The purpose of this paper was to see how the addition of values to a smart reminder system would improve the choice for more appropriately timed notifications. It was reasoned that the most appropriately timed notification would maximally improve user values.

How to increase user values? This increase in user values consisted of two parts: the gains invoked by remembering and the losses invoked by the nuisance of the notification. As actually remembering is mostly determined by the time between the notification and the deadline, the expected value gain caused by the reminder was modelled through this time.

As no sufficing, existing, smart reminder system existed, it was built from scratch. As such, no absolute measure of performance could be introduced. However, a scenario was introduced that would analyze the performance of just the predictive model. Consequently, the value-based approaches could be tested and compared to the baseline to show possible improvements in results. This way, a clear baseline was established and subsequently the effect of values could be comparatively tested. This was further illustrated in Figure 14.

To recap on the concept, the idea is to find a moment, an activity, during which the notification will be presented. To find this moment, for every user, several variables are known and several variables may be calculated from them:

What is known:
- The activities of all users
- The most recent activity
- The QUVs (quantified user values)
- The deadline

What is to be calculated:
- The expected value (probability × QUV) of each possible moment
- The expected time (in steps) between this moment and the deadline
- A score meant for optimization, aimed at maximizing the expected value and minimizing the expected time

For this, the activities of the user are clustered. Given the knowledge of each successive cluster, a predictive model is made. Based on the predictive model and a given deadline, the moment selection algorithm is done according to the following algorithm:

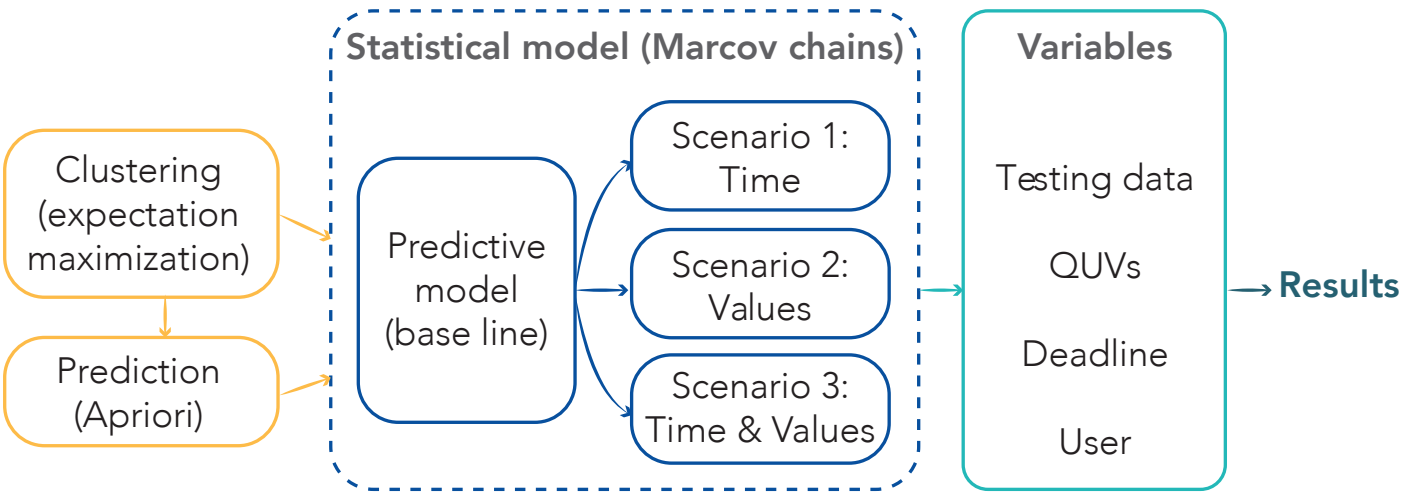

Figure 14 - Approach to testing

Given a current activity cluster A and a goal activity Z, we are looking for an activity cluster S with the highest QUV, that is likely to occur before the deadline, and that will be reached with only a minimal number of expected steps remaining before Z is reached. So:

$$A \rightarrow [n\ steps] \rightarrow S \rightarrow [m\ steps] \rightarrow Z$$

Where the aim is to find a minimal m with a maximal probability and value for S. If the predicted activity cluster S corresponds to the next recorded activity cluster, the notification is dispatched. Else the newly recorded activity cluster is set as the new state A and the process is repeated. This continues until either an S is found, or Z is reached.

With the concept and implementations as discussed in the past chapters, this could now be calculated. However, to test the actual improvements of the concept, a method of testing had to be conceived.

5.2 Method

As previously mentioned, since the smart reminder system was built from scratch, no absolute scoring method could be used. Therefore, a baseline and a relative scoring function had to be established. Thereafter, various testing scenarios could be considered. These three scenarios, as shown in Figure 14, analyze the performance of the algorithm when considering:

- The value loss invoked by the nuisance of the notification
- The expected value gain invoked by remembering as simulated by the time between the moment of notification and the deadline
- Both aspects combined

These scenarios were consequently tested for the different users and deadlines, using a part of the users' activities for training and a part for testing. To simulate random users, the tests were repeated for a large set of randomized QUVs. More explanation and reasoning are discussed in the following sections.

5.2.1 Baseline

Given that the model works with a custom prediction method, it is only possible to analyze results by comparing them with a baseline scenario in which values are not taken into consideration. As the optimization of time between the moment of notification and the deadline was included to simulate the effectiveness of the reminder, and therefore its positive effect on user values, it too was ignored in the baseline. The result is a scenario in which the notification is dispatched during an activity (cluster) which is most likely to occur before the deadline. This corresponds to letting the implementation run with the following scoring function:

$$\text{score}[\text{X}] = P[X]$$

where P[X] is the probability of X occurring before the deadline. Furthermore, the algorithm is considered successful if it manages to correctly predict a moment before reaching the deadline. Obviously, this baseline will have the highest scoring success rate, because it is not being held back due to value or time requirements. However, both the score and success rate should be normalized before comparison. The reasoning behind such a baseline scenario is that it analyzes the performance of the basic algorithms before implementing the value-based decisions.

5.2.2 Scoring and comparing

Like the baseline, every tested scenario will be accompanied by a scoring formula used to calculate a score and a success rate. However, since the different scenarios must be made comparable, two normalized scores are introduced: the normalized score and the normalized success rate.

The normalized score recalculates the score for the chosen activity cluster, but incorporating the value-based aspects. This corresponds to the following function:

$$\text{normalizedScore[X]} = \frac{P[X] * V[X]}{T[X]} = \frac{E[X]}{T[X]}$$

Where V[X] is the QUV corresponding to X as provided by the user, or through random selection (as done in the test runs). T[X] indicates the expected number of steps between X and the deadline. Lastly, E[X] is the expected value of X.

A score of zero indicates a failure, either because no moment was selected, or because the chosen activity cluster corresponded to a QUV of 0. Other than that, the score in itself is fairly meaningless. However, as ideally the expected value is maximized while the time between the moment and the deadline is minimized, a higher normalized score indicates a better performing system.

A good result for a test scenario would be one of three things:

- If the normalized success rate stays roughly the same but the normalized score increases, it means the consideration of the user's value is increase without compromising the effectiveness of the reminder.
- If the normalized score stays roughly the same but the normalized success rate increases, it means an appropriate moment is found more frequently, without compromising user values.
- If both increase, it means all-round, more appropriate moments are found more frequently.

5.2.3 Scenarios

Three scenarios were used to analyze the improvement as caused by including values. Firstly, the scenario in which only time, or the number of steps from the moment until the deadline, is considered. A moment was sought which is an expected minimal number of steps before the deadline, but still likely to occur. In principle, this is still partially related to the predictive model. Therefore, it is interesting to compare this to the other value-based scenarios. For this, the following formula was used to calculate the score of each run:

$$\text{score[X]} = \frac{P[X]}{T[X]}$$

In the second scenario, only values are considered. This was done to see whether incorporating the aspect of time is truly beneficial. This led to the following formula to calculate the score:

$$\text{score[X]} = P[X] * V[X] = E[X]$$

Aside from this, the third scenario takes both time and values into consideration and basically attempts to optimize all aspects. As such its scoring formula is similar to the normalized score.

$$\text{score[X]} = \frac{P[X] * V[X]}{T[X]} = \frac{E[X]}{T[X]}$$

In producing this scoring formula, it was assumed that the value and time components weigh equally. However, for any user these weightings may vary. Since no specific users were analyzed, this was not taken into consideration for now.

5.2.4 Variables

In order to properly randomize tests, a number of variables were altered to create test cases. The four variables were:

- User
- Deadline
- Testing case
- QUVs

Users

As mentioned in 3.2.2, the dataset used has over 6000 data points for four different users, leading to roughly 1500 points per users, spread over roughly a month. The different users were analyzed to prevent any habits of a single user from affecting the results.

Deadline

The deadline, or deadline activity, is the activity before which the notification should have been dispatched. While any activity may be chosen as a deadline, realistically only a few of them create plausible scenarios. For the purpose of this report, the two deadlines as mentioned in Peter's example of having to close the garden doors are: leaving the house and going to sleep. In conjunction with the activities presented in the dataset, this corresponds to the activities 'sleep' and 'outdoors'.

Testing case

In order to perform testing, the datasets have to be split up into a set for training and a set for testing. With an average size of roughly 1500 data points per user, a testing set of 10% should be sufficient [77]. However, as will be short explained, not all of the data points in the testing set will be considered.

The normal procedure would have been to run the algorithm for every data point in the testing set. However, since the data points are not independent of one another, this is not a suitable approach. Instead, testing cases should be considered. A testing case is a continuous series of data points (recorded activities). As many useful testing cases as possible should be identified from the testing set. To select such a testing case, the start and finish of the series of activities should be identified.

The choice for the finish is simple: it is an instance of the deadline activity. The logical choice for the start of the test case would therefore from the prior instance of the deadline activity. Nevertheless, this choice should be reasoned. To illustrate the selection of a testing case, consider the following (simplified) series of activities:

... → ***Sleep*** → *Grooming* → *Eating* → *Outdoor* → *Work* → *Outdoor* →
Eating → *Watching TV* → *Toilet* → *Watching TV* → *Grooming* → ***Sleep*** → ...

Aside from this consider the original example scenario of Peter having to remember to close his garden doors before going to sleep. The most logical choice would be to look at the instance of sleep before it (both marked in bold). As such the prediction algorithm is run. If the predicted cluster corresponds to that of 'grooming', the notification should be dispatched, else the process is repeated. However, the next step would be the same as considering a testing case from 'grooming':

... → *Sleep* → ***Grooming*** → *Eating* → *Outdoor* → *Work* → *Outdoor* →
Eating → *Watching TV* → *Toilet* → *Watching TV* → *Grooming* → ***Sleep*** → ...

As such, given that the reminder was programmed before the testing case in question, it is only necessary to look at the original situation from 'sleep' to 'sleep'. This would be the case for recurring reminders such as Peter's example. Also considering setting reminders at a later moment requires more attention. For example, setting a reminder to call someone before the end of the day. Programming such a reminder is generally done a minimum number of steps before the deadline. Furthermore, this would require a larger resolution than provided by the used dataset. As such, the choice was made to focus on recurring reminders. Furthermore, not every instance of sleeping should be invoked as a deadline. For example, when Peter wakes up to go to the toilet and go back to sleep, this is not a moment at which a notification should occur. Consider this extended example:

... → *Sleep* → *Toilet* → ***Sleep*** → *Grooming* → *Eating* → *Outdoor* → *Work* → *Outdoor* →
Eating → *Watching TV* → *Toilet* → *Watching TV* → *Grooming* → ***Sleep*** → *Toilet* → *Sleep* → ...

Instead a typical testing case would be to find the perfect moment between actually waking up for the start of the day, and going back to sleep. In other words, the testing set would be the set of activities between the two marked instances of 'sleep'. In order to find such testing sets it was assumed that such 'connected' activity instances may happen with a maximum of two different activities in between.

QUVs

In order to objectively test the model with respect to the QUVs, or values relating to annoyance caused by the notification, the QUVs should be randomized. As such, every scenario was tested with 1000 different configurations of values. Here, each random value was an integer value between 0 and 4, linked to an activity corresponding to the values as mentioned in 3.4.3.

Number of runs

Given that there are 4 users, 2 deadlines and 1000 random sets of values, a minimum of 8000 results are obtained given that there is at least one test case per user per deadline. Realistically, a much higher number is achieved. Using 20% of the dataset as testing activities led to roughly 24000 results.

5.2.5 Implementation

As mentioned before in 5.1.2, for the various scenarios, four numbers were calculated: the score, success rate and their respective normalized values. Performing this for all users, and randomized variable led to just under an hour of testing. This long testing time was the result of only using semi-optimized.

5.3 Results

Before looking at the results, let us quickly recap the meaning and purpose of the normalized results. While the opportune moment was calculated using the formulas corresponding to each scenario, the moment is reevaluated while taking all variables into consideration. Consider the following example for illustration:

> *In the baseline scenario where values and time are not considered, the appropriate moment is found to be at activity 'work'. However, the user considers it 'unacceptable' to be notified during the activity. Hence, its QUV and respective normalized score are 0.*

Here a successful moment in the baseline scenario would be a very unsuccessful moment in the eyes of the user when considering their values. As such, normalized simply means, 'the score or success rate as it would have been taking both values and time into consideration', allowing comparison between the different scenarios.

The success rate is simply the fraction of tests which successfully predicted a moment suitable for notification according to the scoring function. When normalized, it is checked against the QUV corresponding to the chosen activity (if this had not yet been done). Hence, the basic and normalized success rates will be equal for scenarios already including QUVs in its scoring function.

Tables 2 and 3 show the combined results for 24015 individual test runs for the various variable and scenarios. s indicates the respective score or success rate. Δs indicates the difference in regards to the baseline, however this is not applicable to the not-normalized scores.

Scenario	Baseline	Only time		Only values		Values & Time	
	s	s	Δs	s	Δs	s	Δs
Success rate	0.916	0.833	-9.1%	0.828	-9.6%	0.776	-15.2%
Normalized SR	0.733	0.667	-9.0%	0.828	+13.0%	0.776	+5.8%

Table 2 - Success rates (SR) per scenario

Scenario	Baseline	Only time		Only values		Values & Time	
	s	s	Δs	s	Δs	s	Δs
Score	0.917	0.077		3.083		0.231	
Normalized Score	0.195	0.195	0%	0.227	+16.4%	0.231	+18.5%

Table 3 - Scores per scenario

From these results several interesting conclusions may be inferred. Firstly, the baseline is shown to have a success rate of roughly 92%. This is actually a very good results since it shows that the predictive model works quite adequately. The normalized success rate of 73% (which is roughly 20% lower than 92%) indicates that the randomization of the user values properly works. Given that there can be 5 possible values, one of them being 0, 20% should indeed fail.

Furthermore, in all other scenarios, the addition of time or values in the calculations is shown to have a negative effect on the success rate. This is to be expected given the introduction limiting the number of appropriate moments. However, compared to the normalized success rate of the baseline, there is actually quite a large improvement for the value based scenarios. This shows that the addition of values in a model actually improves the selection of an appropriate moment for notification in terms of successful selection. The fact the inclusion of time only has a negative effect on the normalized success rate can be explained by the fact that it is only a limiting factor but without a cut-off (such as the 'unacceptable' QUV). Ultimately, the most important aspect is that unless the user demands much emphasis on the timing of the notification, the normalized success rate shows clear improvement.

The next aspect is to consider the normalized scores. Obviously, trying to optimize the same function used to calculate the normalized score will give the highest result. More interesting, are the three scenarios side by side. The inclusion of just time does not show any (significant) improvement. The inclusion of just values does so drastically. This can be explained by the many 'unacceptable' activities no longer being considered as an appropriate moment. However, the kicker lies in the improvement from this to including both the values and time components. This shows that, while still improving the success rate, the addition of user values in a model actually improves the selection of an appropriate moment for notification.

While this score can probably be improved even further, this is mostly dependent on the preferences on the user. The user preferences alters weighting between the values, timing, and success rate.

5.4 Conclusions

The basic prediction algorithm has shown to work properly, showing a success rate of almost 92%. Also considering time and value components, the optimized scenarios have shown improvements of 6 – 13% successful predictions. In terms of more appropriate moments, the optimized scenarios score up to at least 18.5% better, without losing too much on the success rate. Further improvements may be attained by proper weighting of the values, timing, and success rate, based on the user's preferences.

5.4.1 Limitations

While the results are promising, a number of assumptions had to be made, introducing a number of limitations. Reverting these limitations may have a negative effect on the results.

Optimization to time

Rather than looking at continuous time, discrete time was considered, counting activities rather than the duration of the activities. Assuming continuous time

No cut-off time

Whether the time before a deadline is an accurate measure of the probability of actually remembering is a non-trivial assumption. Instead a cut-off time could be used.

One value

The implementation focuses on a single value: the nuisance invoked by the notification. This is done as a reasoned replacement of other user values. Once again this is not a trivial assumption, albeit a very functional one. However, the concept can be easily extended to support various user values by doing all calculations for said values and weighting them to how the user cares about the different values. This would, however, require vast amounts of user input regarding values.

Experimental domain

It was assumed that the system works with reminders that come periodically. As such it was not tested for reminders set relatively shortly before its deadline. Therefore, the applicable domains are currently more suited for elder care (as suggested by the main example of Peter). However, this does not mean this cannot be easily expanded.

Limited Apriori sets

As mentioned in 3.3.4, a choice was made to follow the paper by Nazerfard et al [37]. When computing the Apriori sets, only transactions of two subsequent activities were considered. However, the power of the Apriori algorithm, as well as other predictive algorithms, is that it is aimed at identifying larger sets. In other words, sequences of activities likely to follow one another.

A possible way of expanding the sets while still being able to use Markov chains for the statistical model is to view every set as a single state in the Markov chain. However, the difficulty lies in the mathematical implications this will have on the further calculations.

Multiple possible deadlines

While the implementation perfectly allowed for multiple deadlines, a choice was made to only allow for a single deadline activity at a time. The reason for this was to increase understanding of the results. However, this limitation can be surpassed by adding another absorbing state to the transition matrix and slightly altering the dimensions of the resulting matrices.

6. Conclusion & Discussion

The purpose of this paper was to see how the addition of values to a smart reminder system would improve the choice for more appropriately timed notifications. Having concluded the project and having seen the results, this main question, along with its sub-questions may be answered.

The goal of this report was to answer the question:

How can a smart reminder system be extended to incorporate user values to provide more appropriately timed notifications?

The main concept considered throughout the paper was to extend a predictive smart reminder system with the concept of user values. The goal? To find a more appropriate moment than simply setting a timer, or even just utilizing the predictive model. For this, existing such systems were analyzed and compared to see whether they were suitable for extension.

What are the requirements for the smart reminder system?
Most importantly, the user's activities should be represented in the model. This could be done through the use of a predictive algorithm. The result of this model should be a list of probabilities or scores of the activities that can subsequently be combined with values. Further requirements are the inclusion of the concepts of goals as well as the model dynamically adapting to the user's current activities.

Which existing models and systems exist for smart reminder systems and how do they compare?
Actually, very few such systems exist and if they do they are very limited in their functionalities. Systems may incorporate selective data about the user's activities but, for example, only use it to find moments when they are not working. Most other systems do not work dynamically and instead user environmental data, such as geofencing, to plan reminders. Value based design is generally not included anywhere other than at design time.

How can user values be analyzed and quantified?
There are many different ways of looking at user values. This is in part due to the large number of possible values. First and foremost, a selection has to be made as to which values are considered. Further difficulty lies in that there is no clear way to compare different values other than quantifying the values and quantifying the importance of the values.

To simplify matters, rather than looking at the different values, only a single value is considered: the nuisance caused by dispatching the notification at a specific activity. This facilitates further calculations and eliminates uncertainty due to vague comparisons between values that might be weighted differently by the user.

Furthermore, the assumption is made that a longer time between the reminder and the deadline reduces the value of the reminder. As such, two value-based aspects are considered in the ultimate concept. The value loss due to the notification and the value gain caused by the reminder, as simulated by the time between notification and the deadline.

How can the model be extended to incorporate user values?
Once a prediction model is made, its probabilities can be combined with quantified values. Consecutively a statistical model can be used and optimized to find the most appropriate time for dispatching a notification.

Since no sufficient implementations were found to exist, a concept was designed and implemented from scratch. To limit the scope, no sensor analysis was done, but instead a dataset was used containing clear information about a user's ADL. Further arrangements were made to also allow data streams from third parties. Using expectation maximization and Apriori algorithms as respectively clustering and prediction methods, a statistical, predictive model can be established in the form of a Markov chain. The properties of the Markov chain are then used to identify the expected value of each possible activity (or rather activity cluster) and their expected time remaining until the deadline. Ultimately, these two values are combined into a score which is optimized.

How should the smart reminder model be implemented in order to allow easy collaboration with third party software?
A complete Node.js web application with RESTful API was decided upon due to its clear structure, its ability to work in the cloud its openness in connecting to other pieces of software and IoT devices. This platform was also used to implement the design and its algorithms.

A separation of concerns was done by splitting the implementation into frontend, resource handling through the API, and the actual methods used for calculations and predictions. After collection of the activities, the algorithms could be run to create a prediction model. Combining the prediction models and the user values, the calculations could be performed to find the most appropriate moment. This gave a glimpse into how a fully functional application could work. Furthermore, it allowed for

future collaboration with other smart systems as well as easy creation of test scenarios.

Does the use of the value-extended model provide more appropriately timed notifications?
The results show that the predictions which incorporate user values into its decision model provide more appropriately timed notifications in comparison to the predictions that ignore user values. An improvement of 6 – 13% was observed in successful predictions, with the predictions predicting up to at least 18.5% better scoring (more appropriate) moments. This shows a clear improvement over the basic predictive model, indicating a clear advantage to using a value-based system. Whether this is more appropriate than other smart reminder models would require further testing. However, this approach to appropriately timed notifications is, first and foremost, a feasible one. Which directly answers the main research question:

How can a smart reminder system be extended to incorporate user values to provide more appropriately timed notifications?
Through the use of quantification and statistical models, any predictive model could be extended with the concept of user values and attain useful and improved results.

6.1 Scientific and practical contribution

The scientific contribution of this project was its clear concept of extending a predictive model with user values in a dynamic and statistical manner. It showed how the incorporation of user values could be used to improve planning of notifications. It has shown how several existing concepts can be combined to create a complex and dynamic model. While still in a rudimentary state, it is directly usable and prepared for various paths for further research.

The practical contribution was, through the actual finished implementation. Through designing, creating and testing an implementation, it is directly interesting for use in corporate applications. Companies that already work with planning and activity information could benefit from its applications.

6.2 Future enhancements

There are several aspects which warrant closer inspection when revisiting this project.

6.2.1 Differentiating between values

In 3.4.2, the choice was made to look at a single value. While appropriate for this implementation, it does limit the way in which values can be considered. Firstly, notifications may invoke losses in several different values. Furthermore, different values may have a different level of importance to users. Comparing these differences may provide more insight into the effects of the values on the ideal moment.

Similarly, the value of remembering should be taken into consideration. While the assumptions regarding this matter are appropriate, there is one case that is not being considered. That case occurs when the notification incurred loss is always higher than the value gain invoked by actually remembering. This raises questions as to whether the reminder should be planned at all.

Both changes would be very interesting. They would, however, drastically increase the complexity of the mathematical calculations needed to be performed.

6.2.2 Clustering based on more parameters

As mentioned in 3.3.2, a choice was made to follow the paper by Nazerfard et al [37]. In this, prior to doing activity prediction, activities were clustered based on their starting times. Subsequently, their durations were used to exclude outliers from the cluster calculations. However, the durations could instead be very well used within the clustering itself. Rather than clustering based solely on starting time, duration could be added as a second parameter. Similarly, other parameters could be introduced.

6.2.3 Goal reasoning

As mentioned in 3.4.2, rather than applying the concept of goal reasoning as described in [34], attaining the goal was made synonymous with arriving at a certain activity. In reality, attaining a goal is much more dependent on a number of prerequisite activities.

An initial idea for this would be to look at the larger Apriori sets as mentioned just before. However, these prerequisites do not necessarily have to be completed in order. As such, more research would

be required in order to implement this. Most likely, a solution could be found through combining the concepts from [34], [42].

Another aspect is that there can be more than one activity related to the goal. In the main example of Peter, two goal activities were mentioned: Sleeping and leaving the house. While the current implementation allows for only a single goal activity, there is nothing that blocks expansion to multiple goal activities. This is done by simply making both states absorbing and adjusting all calculations accordingly. While demanding a bit of time, it is not at all an unattainable next step in improving the concept of this paper.

6.2.4 Improving Other prediction methods

The prediction methods based on clustering and the Apriori algorithm are definitely not the most efficient or the most accurate. They are, however, acceptably accurate and easy to implement and tweak. With more and more advanced machine learning algorithms being developed, upgrading the implementation of this paper with such a prediction method would be an interesting undertaking.

6.2.5 Analyzing user preferences

The scoring methods used in the prediction algorithms consider the user to equally weigh the user values, time until the deadline and the success rate. These three parameters can all be optimized based on the preference of the user. Ideally, the current system is used as a basis. Then based on user feedback, the weightings are adjusted. Or, if a substantial amount of feedback had already been collected, the average user weightings may be used as a starting point.

6.3 Final remarks

The concept and implementation as presented in this report provide a clear and adequate basis. Numerous improvements and changes can be made to increase the effectiveness of this solution. Nonetheless, it has clearly been shown that through the use of quantified values and a statistical model, any reminder system or predictive model can be made aware of said values and use them to generate notifications in a more user centric manner.

9. References

[1] "Too dependent on technology," *The Nation*, 31-Dec-2017. [Online]. Available: https://nation.com.pk/01-Jan-2018/too-dependent-on-technology. [Accessed: 02-Feb-2019].

[2] 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.

[3] 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.

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

[5] 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.

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

[7] 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.

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

[9] K. MorrisonlMarch 12 and 2015, "Needy Technology: Too Many Notifications Causes Users to Tune Out." [Online]. Available: https://www.adweek.com/digital/needy-technology-too-many-notifications-causes-users-to-tune-out/. [Accessed: 02-Feb-2019].

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

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

[12] "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].

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

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

[15] A. Robertson, "Location/time-based reminder for personal electronic devices," 06-Dec-2000.

[16] Jason F. Hunzinger, "Location specific reminders for wireless mobiles," 15-Nov-2001.

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

[18] 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.

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

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] "State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating." .

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

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

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

[29] "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].

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

[31] 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.

[32] 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.

[33] 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.

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

[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] 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.

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

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

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

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

[41] P. Pasotti, C. M. Jonker, and M. B. van Riemsdijk, "Towards a formalisation of Action Identification Hierarchies*."

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

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

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

[45] 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.

[46] 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.

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

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

[49] 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.

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

[51] 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.

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

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

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

[55] 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.

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

[57] 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.

[58] A. Tversky and D. Kahneman, "Advances in prospect theory: Cumulative representation of uncertainty," p. 27.

[59] H. N. Boone and D. A. Boone, "Analyzing Likert Data," *J. Ext.*, vol. 50, no. 2, Apr. 2012.

[60] "Multitasken is toch niet zo handig," *Psychologie Magazine*, 07-Nov-2013. .

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

[62] 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].

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

[64] 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.

[65] M. Wilcox, S. Schuermans, C. Voskoglou, and A. Sobolevski, "State of the Developer Nation," 2017.

[66] "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].

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

[68] "npm." [Online]. Available: https://www.npmjs.com/. [Accessed: 12-Jan-2019].

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

[70] "Mongoose ODM v5.4.3." [Online]. Available: https://mongoosejs.com/. [Accessed: 12-Jan-2019].

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

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

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

[74] "What is a RESTful API?," *MuleSoft*, 02-Oct-2017. [Online]. Available: https://www.mulesoft.com/resources/api/restful-api. [Accessed: 13-Jan-2019].

[75] Assertible, "7 HTTP methods every web developer should know and how to test them," *Assertible*. [Online]. Available: https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them. [Accessed: 13-Jan-2019].

[76] K. Sera, *Apriori Algorithm implementation in TypeScript|JavaScript: seratch/apriori.js*. 2018.

[77] I. Guyon, "A Scaling Law for the Validation-Set Training-Set Size Ratio," in *AT & T Bell Laboratories*, 1997.

8. Appendices

8.1 Key concepts of researched papers.

AHCS/TAFETA [22], [5]

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

Gate reminder [18]

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

PAIR [3]

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.

CogKnow [20]

This concept is one that implements user values, except not in the way that is desired in this project. Instead, it 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.

Decision maker [32]

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

Goal models [34]

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

Olisto/IFTTT/CAMP/CybreMinder [10], [11], [27], [28]

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 [31]

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

MLCARS [30]

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

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.

Attelia [2]

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

What should I do/Action Hierarchies {40], [41]

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

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

MagHive [29]

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

Fuzzy linguistics [33]

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.

Long term evaluation of smart homes [36]

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 [37]

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

8.2 Dataset entry

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 shown in figure XXX.
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.

```
<event>
	<string key="concept:name" value="sleeping" />
	<string key="lifecycle:transition" value="start" />
	<date key="time:timestamp" value="2012-11-14T05:16:51.000+01:00" />
	<string key="Column_4" value="sleeping" />
</event>
<event>
	<string key="concept:name" value="sleeping" />
	<string key="lifecycle:transition" value="complete" />
	<date key="time:timestamp" value="2012-11-14T09:01:27.000+01:00" />
	<string key="Column_4" value="sleeping" />
</event>
```

Figure 15 - Typical entry for both events

8.3 Unique activities in dataset

bathe	mealpreperation	sleep
cleaning	medication	snack
dress	outdoors	toilet
drink	personalhygiene	watchtv
eatingdrinking	phone	work
entertainguests	read	
groom	relax	

8.4 Platform

The chosen platform is not just dependent on the chosen algorithm 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.

8.4.1 Background

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 [25]. 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 [62]–[64], 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.

8.4.2 Programming language

When it comes to implementing machine learning algorithms, there are several go to languages. The five most used languages [65], in order, are:

- Python
- C/C++
- Java
- R
- JavaScript

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.

8.4.3 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% [66]) is attributed to Node.js webservers.

Software platform – Node.js

JavaScript was originally a client-side scripting language, running in the user's browser, usually part of any website. Node.js [67] 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) [68] 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 [69] 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.

Mongoose [70] provides a straight-forward, schema-based solution to model application data as it is stored in a MongoDB type database (described later in 4.2.4). 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.

The ml.js suite [71] 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 3.

View engine – Handlebars

The application needs to, among other things, handle user input and allow for datasets 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 [72] 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 [73]. 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.

8.4.4 API

An application programming interface, or API, is a collection of definitions used among applications to communicate between one another. More complex code is abstracted for simpler use. Rather than having an application know all low level details of the platform on which it is running or the library it is using, it allows it to use predefined building blocks. APIs are generally used in libraries, operating systems, web services and many other implementations.

Take a printer, for example. When you click the print button in an application like Microsoft Word, it is not this application that knows how to drive a printer. Instead, it calls a function in printing API in the underlying operating system. The operating system can, in turn, invoke the printer driver to print the document.

Web APIs

Web APIs, is an API used over the web, that can be accessed via HTTP requests. It is used as an interface between a service and a client application which uses its assets. Within the definitions of the API are properties such as hostname, path, query parameters, error codes, etc.

For the purpose of this project and its implementation, such an API facilitates a number of matters. Firstly, it allows for a clearly structured approach to handling and communicating data. Secondly, it allows the frontend, the client-side webpage, to fetch information such as statistics while also being able to provide possibilities of uploading data such as new datasets and user value information. Lastly, it provides a way for other services to connect with it.

To illustrate the last point, the most obvious example is the option to facilitate a data stream. Subscribing to such a data stream is generally done through the concept of webhooks. In its most simplest form, service A sends a request to service B to subscribe to certain events. Whenever such an event happens

at service B, it sends a request to service A with the information regarding the event.

RESTful API

A RESTful API is one based on representational state technology (REST). This is a standardized, architectural approach web communication using HTTP methodologies.

A main advantage of a RESTful API is that it provides a great deal of flexibility. Because data is not tied to methods or resources, multiple calls can be handled simultaneously, different data formats can be returned, and like these there are many more advantages. This flexibility allows developers to build an API that meets meeting all kinds of needs [74].

When designing such an API it is important to understand its concepts and constraints. Firstly, the API should be stateless. This allows calls to be made independently from one another. As such, each call should contain all data necessary to execute successfully. Secondly, the API should be designed with the concept in mind that the server and client are distinct and should be able to evolve separately from another. Lastly, resulting from the previous point, the API should have a uniform interface. In this way, the services are not tightly coupled to the API itself. In order to achieve this, where applicable, each resource should implement the HTTP methodologies properly, rather than using random endpoints. Each resource, such as an activity, cluster or user, should be accessible through these methods. The most common methods (and the only ones used in this implementation) are [75]: GET, POST and DELETE. They are used to respectively retrieve, create or update, and delete a resource.

8.4.5 Conclusions

Combining all the aforementioned aspects, in order to allow for easy collaboration with third-party software, a JavaScript based, Node.js server was established, along with a MongoDB database. It serves a frontend used to view statistics and allows for user input. All communication is done through a RESTful API. For performing complex machine learning calculations and matrix calculations, several library functions were imported.

Master thesis
Remy Kabel
TU Delft