

Slovak University of Technology Bratislava  
Faculty of Informatics and Information Technologies

FIIT-5220-47788

Bc. Anton Benčíč

INFORMATION RECOMMENDATION USING  
CONTEXT IN A SPECIFIC DOMAIN

Master Thesis

Supervisor: prof. Ing. Mária Bieliková, PhD.

2012, May



Slovak University of Technology Bratislava  
Faculty of Informatics and Information Technologies

FIIT-5220-47788

Bc. Anton Benčíč

INFORMATION RECOMMENDATION USING  
CONTEXT IN A SPECIFIC DOMAIN

Master Thesis

Degree Course: Software Engineering

Study Line: 9.2.5 Software Engineering

Place: Department of Informatics and Software Engineering, FIIT STU  
Bratislava

Supervisor: prof. Ing. Mária Bieliková, PhD.

2012, May



# Anotácia

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLOGIÍ

Študijný program: SOFTVÉROVÉ INŽINIERSTVO

Autor: Bc. Anton Benčíč

Diplomová práca: Odporúčanie informácií s využitím kontextu v špecifickej doméne

Vedúci diplomovej práce: prof. Ing. Mária Bieliková, PhD.

máj, 2012

Personalizačné a adaptívne metódy vo všeobecnosti sú súčasťou procesu odporúčania. Tento proces sa skladá z niekoľkých rozhodnutí, ktoré musia byť urobené aby sa v určitom čase a určitým spôsobom doručil používateľovi určitý obsah v určitom množstve. Veľmi často sa však odporúčanie zaoberá iba na jednom z týchto rozhodnutí, a to najčastejšie býva práve rozhodnutie týkajúce sa obsahu. Toto platí špeciálne pre pasívne metódy, ktoré dodávajú obsah na požiadanie a nemusia sa teda starať o to, či je vhodný čas priniesť používateľovi vybraný obsah, aké množstvo je správne a ako tento obsah používateľovi prezentovať, čo sú ďalšie rozhodnutia prítomné v procese odporúčania.

V tejto práci sa sústreďujeme najmä na proaktívne metódy, ktoré sa samé rozhodujú pre vykonanie akcie a svoje rozhodnutia zakladajú na rôznych typoch kritérií. Takouto akciou môže byť jednoduché nastavenie zvukového profilu telefónu na základe definovaných časových okien alebo niečo tak komplikované, ako odporúčanie hudby pre používateľa a priateľov v jeho okolí podľa ich nálady.

Náš projekt sa zameriava na návrh a realizáciu metódy, ktorá sa dokáže účelne a účinne naučiť aké akcie majú byť vykonané v špecifických situáciách a následne využiť tento model pri odporúčaní akcií pre tieto situácie. V práci okrem všeobecného návrhu taktiež opisujeme aplikáciu metódy v doméne internetových noviniek a jej vyhodnotenie využitím vytvoreného simulačného prostredia ako aj výsledkov živého experimentu.



# Annotation

Slovak University of Technology Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree Course: SOFTWARE ENGINEERING

Author: Bc. Anton Benčíč

Master's Theses: Information Recommendation Using Context in a Specific Domain

Supervisor: prof. Ing. Mária Bieliková, PhD.

2012, May

Personalization and adaptation methods in general engage in a recommendation process. This process consists of a few decisions that have to be made in order to deliver something specific to a user. More often than not adaptive and personalization methods engage in only one decision and that is what to deliver to the user. This is especially true for methods that work in an on-demand basis, thus deliver content when a query for it is made. These methods generally do not consider if it is the right time to deliver the content, in what volume should it be delivered and how should it be presented, which are the other decisions present in the recommendation process.

In this work we concentrate mainly on methods that work proactively. A proactive method decides on an action by considering various types and amount of criteria. This can be as simple as setting the sound profile according to the user defined time windows or as complicated as recommending the right music for a given user and her friends around considering their mood for example.

Our project is aimed at designing a method that is able to effectively and efficiently learn what actions should be performed in what situations and then use this model to recommend actions given a particular situation. Besides the general method's design we also describe our method when being applied to the domain of internet news and evaluate it using both our simulation framework and results gathered from the live experiment we performed.





# Table of Contents

1. Introduction .....	1
2. Information Recommendation Process .....	3
2.1 Context and Situations .....	4
2.2 Proactive Information Recommendation .....	6
2.3 Use of Context in Information Recommendation .....	9
2.4 Context-Awareness Problems .....	12
2.5 Privacy Concerns .....	15
3. Domain of Internet News .....	17
3.1 User Behaviour Observations .....	17
3.2 News Recommendation .....	21
4. Method for Action Recommendation Based on Situation Rules.....	23
4.1 Situation Model.....	23
4.2 Action Model.....	28
4.3 Rule Generation .....	31
4.4 Recommendation Calculation .....	33
4.5 Method Characteristics and Discussion .....	38
5. Method for Action Recommendation in the Domain of Internet News .....	41
5.1 Situations .....	41
5.2 Actions and Indicators .....	45
5.3 Domain Characteristics and Discussion.....	46
6. Evaluation of the Method for Action Recommendation .....	49
6.1 Base Simulations .....	49
6.2 Live Experiment.....	55
6.3 Additional Simulations.....	57
7. Conclusions and Future Work .....	61
References.....	63

Resumé v slovenskom jazyku (Resume in Slovak Language)

Appendix A: Technical Documentation

Appendix B: Reference Manual

Appendix C: Paper Submitted to RecSys 2012

Appendix D: Journal Article Draft

Appendix E: IIT.SRC 2012 Poster

Appendix F: Contents of the Digital Medium

# Chapter 1

## Introduction

Information recommendation in its basic form consists of a three step process. The first step is deciding when to recommend some content. Then we have to decide what kind of content should be recommended and in what volume, and finally how to present it to a user. None of these decisions is trivial when we want to perform it in an intelligent way. Most of existing recommenders focus solely on the content part considering others constant which is acceptable, as long as we are dealing with on-demand recommendation. In other words recommendation that takes place after a query of sorts, be it sorting of search results or recommending additional products in a web shop. In these cases we do not have to worry about the time of recommendation as someone else, most often a user, makes the decision.

On the other side in our world of pervasive computing we can see more and more attempts for intelligent proactive recommenders. A proactive recommender is one that engages in the first step of the recommendation process - deciding when a recommendation should happen. This timing decision that is generally based on situation identification often influences all of two three other stages as there is an interdependency link between them. For example the content provided by an intelligent news push service should be different when at a sports match with our friends than the content provided after a movie with our significant other. This goes for content volume and means of presentation as well.

Our main aim is to design and develop a method that is able to efficiently recommend actions based on a user's situation. We aim for final design that is easily portable among various domains and helps end-user service or application designers to include context awareness with little or no overhead.

Our work is structured as follows. In Chapter 2 we introduce the information recommendation process. We start with describing the distinction between on-demand and proactive recommendation methods and introduce the situation and context concepts with it. We continue by outlining what aspects can be present with proactive recommendation and discuss them individually in

more detail. After the basics are covered we present some example use of context awareness in information recommendations and also describe the problems that it brings. We conclude the chapter with discussion about privacy issues when working with sensitive context information.

In Chapter 3 we introduce internet news recommender services that we use as our first target domain. First we present some user behaviour observations that we have come through or performed ourselves and discuss on how the results impact method design and realization in the chosen news domain. We also cover a few news recommendation services that already exist and show the concept of situations and actions in their context.

In Chapter 4 we introduce our rule-based method. We start with describing two models used within our method and show how they relate to each other. We continue with detailed description of how the rules are generated and how the recommendations are calculated. We conclude the chapter with discussion about the main characteristics of our method.

In Chapter 5 we outline a setup for using our method in the domain of internet news. We define the situations and actions that should be covered within the models and conclude the chapter with discussion about possibilities and limitations of our method in internet news provision and similar domains.

In Chapter 6 we describe how we evaluated our method. We start with presenting results from our simulation framework that we built for testing our method's basic and advanced characteristics. Then we continue on with a description of short-term live experiment that we performed in order to better approximate how our method performs in the real world. We conclude the chapter a final set of simulations that are based on the findings from the live experiment and a discussion on the results.

We conclude our work with overall evaluation of our method's capabilities and limitations and outline a few directions for our future work.

## Chapter 2

# Information Recommendation Process

When we think about information recommendation based on how is it initiated we can easily distinguish between two types:

- On demand recommendation
- Proactive recommendation

On demand recommendation is undoubtedly the most common type of recommendation currently present in content provision services. This includes for example complementary article recommendations on an internet news site, recommended products in an online shop or even personalized results of an internet search engine. All of these have the common characteristic of being presented on demand, alongside some other user action, content or query.

On the other hand proactive recommenders take the responsibility to inform the user about interesting content or a piece of information on their own. An example of a proactive recommender or search engine that exploits context information is presented in [1]. The search engine tracks user's activities and environmental context to automatically prepare relevant documents for her. Three example agents are introduced in the paper that suggest documents based on what is the user typing in a text processor, the contents of currently loaded web page and information from physical environment accessed by a means of a wearable computer with a set of sensors.

The proactive recommenders are of our main interest so we cover this topic in more depth in the rest of this chapter. There are three aspects related to proactive content recommendation:

- Time of recommendation
- Content to recommend
- Presentation of the recommended content

All of these three aspects form a single coherent unit where one aspect depends on all other aspects and failure to understand this dependence may lead to

substantial problems with the recommendation process and its results. To fully understand the three aspects in proactive recommendation we first have to introduce the concepts of *situation* and *context* and the way we understand them in our work.

## 2.1 Context and Situations

One of the definitions in [2] states that situation is a position of a person with regard to circumstances. In our view the circumstances characterize a state of a user or an environment she is in. This can be for example what the place, time or weather is for a particular user or even what is she doing or who is around.

Regarding the context there are many definitions of what it actually is, but in general context can be anything that describes a situation. According to [3] and then [4] context can be divided into four categories:

- Computing context – such as network connectivity, communication costs and communication bandwidth, and nearby resources such as printers, displays and workstations
- User context – such as user’s location, nearby people or the current social situation
- Physical context – such as lighting, noise levels, traffic conditions or temperature
- Time context<sup>1</sup> – such as time of a day, week, month or season of the year

This is one of the baseline views on context categorization, but there are other views as well. One for example focuses on how can be context information accessed (access view) or another that is more concerned with what is the content of the context information (content view). In access view we distinguish two categories of context:

- Sensor context
- Service context

The sensor context encompasses context information that is accessed directly using device sensors like location, movement (through accelerometers) or noise levels. The service context encompasses information that is accessed using a service like weather, calendar events or user set alarms.

The view that we are most interested in however is a kind of content view. One of the main reasons is that it can be used to categorize context

---

<sup>1</sup> Time is a part of the physical context, but it is often set aside due to its significance

information according to its use in individual stages of the recommendation decision process. The content view contains three categories of context information:

- Device context
- User context
- Background context

The device context includes such information as battery level, screen resolution, processing power or connection quality that describes the device and not the user. This information is mostly valuable for context-aware applications that focus more on the presentation aspect than on what to recommend (content) or when to recommend (time). For example if an application detects slow connection it can opt for a simple text-based interface instead of a rich multimedia one that is much more data intensive. Such scenarios that are mostly related to the presentation stage call for an explicit, non-intelligent situation to action associations and thus are not of our primary focus.

The user context and background context on the other hand describe the situation that the user is currently in and thus are of our primary concern. They are sometimes listed separately, but the difference between them comes down to what the context describes more. In case of user context the information is about the user herself like her current location, fatigue or even mood. In case of background context the information relate primarily to the state of the environment the user is in like weather, noise levels or what event the user is at. Because these nuances are not that important for the purposes of our project we further refer to them as to user and background context or simply user context. Here are just a few examples of commonly used user context information:

- |                             |                     |
|-----------------------------|---------------------|
| • Date and time             | • Stored documents  |
| • Location                  | • Browsing history  |
| • User's calendar           | • Current weather   |
| • User's multimedia gallery | • Nearby facilities |
| • Social feed               | • Nearby people     |

Even though some of this context information like stored documents or multimedia gallery content exhibit more long-term nature and are thus part of the user's profile we can still follow changes and deduce the short or medium-term context from them. Other information like nearby people or current user's location is more short term by its nature and if present can be often used directly. The long-term context information is more suitable for choosing the right content type or means of presentation while the short-term context is what we are most interested in as our main aim is to explore the possibilities in

personalized recommendation in terms of finding suitable situations for particular actions.

## 2.2 Proactive Information Recommendation

With the concepts of situation and context covered we can continue on with analysis of the three aspects related to the proactive information recommendation.

### 2.2.1 Time Aspect

Proactive content recommendation services have to recognize when it is the right time to provide some content or a piece of information and act upon it. In the most trivial case these can be times explicitly defined by the designer. For a news service this may be a few minutes after the alarm goes off, a few minutes before lunchtime or in the evening after the lights in the room are turned off. Another example may be a movie rental service recommending movies on weekend evenings as there is barely anything on TV at that time.

These are however only trivial examples that feature “one size fits all” approach and ignore any differences that exist between individuals. To be fully able to choose the right time for an action we need to exploit the concepts of situation and context as we defined them in Section 2.1. The right time can be then defined as the right or specific situation because it is the situation, not necessarily only time itself that can tell us what to do.

For example imagine a user who gets up four days in a row at the same time and checks the traffic information while leaving home also roughly around the same time. This does not mean that the time is the reason but rather she leaving home should be considered a relevant situation. This way in case of her timetable change when we use the time as our lead we fail to correctly provide content while by choosing situation of her leaving home we are going to be much more accurate.

The main goal here contrary to the presented idea of explicit times is to identify situations or a set of circumstances that are suitable for a specific action. This is just a brief example introduction to situations and their use within the time aspect of a proactive information recommendation and more is covered in Section 2.3 that is devoted to use of context and context-awareness in general.



### 2.2.2 Content Aspect

The content aspect groups both the type of content we are going to recommend as well as the amount of content recommended. When it comes to type of the content provided, recommendation services should certainly do their best to accommodate themselves to the situation at hand. When we recognize that a user is on her way to a tennis match, she is probably going to appreciate sports news much more than technologically oriented ones. On the other hand these may be more suitable for her a few minutes before she is going to have a lunch with her colleagues. This is a simple example of how the situation aspect can to a considerable extent influence the content aspect.

Considering the content aspect without the context there are three basic types of recommendation methods:

- Content-based recommendation methods
- Collaborative recommendation methods
- Hybrid recommendation methods

Content-based recommendation methods analyze item descriptions to identify items that are of particular interest to the user [5]. An item description in this case may be a whole article text, tags assigned to it or anything else that describes the content. As noted in [5] there is a variety of algorithms for learning user profiles from both explicit and implicit user feedback on individual items.

Collaborative recommendation methods on the other hand evaluate individual items based on the match between user's feedback on items and feedback of other users without considering the content of these items. The items in case of collaborative recommendation thus figure as a kind of black boxes. As presented in [6] collaborative recommenders are suitable for a variety of recommendation tasks, mainly where we are unable to understand and grasp the content in its whole complexity correctly due to software and hardware limitations.

Hybrid recommendation methods are essentially structured sets of content and collaborative recommenders used together or in dependence one on another. In [7] there are 53 different hybrid method types shown, each of which combines different methods in different ways and has therefore different characteristics and use. All-in-all the main purpose of hybrid recommenders is usually to combine the best from multiple methods and mitigate the weaknesses they otherwise present individually.

Context based content recommenders also exist. In [8] the authors extend a basic Bayesian feedback classifier with additional dimension for context in

which the rating was assigned. The reasoning behind this is that user opinions consist not only on their interests, but are also formed by the current context. In other words the fact that a user gives negative feedback for some piece of content in a specific context does not necessarily imply that this is true in all other situations. In [9] the authors propose a collaborative service recommender that extends models of user preferences with context information about location, time, visit reason and information about the relationship to the user's travel companions.

One may argue that the volume of the content provided is not that important, but we think it is. Let us consider the exact same situation as with the recommended content type. When we recognize a user going to a tennis match, she probably wants to be informed about as many outcomes of other matches as is technically and mentally possible. The content is usually not that important as the headlines and even though the user may only read headlines of most of the articles, they may all be relevant and valuable to her at the time. On the other hand a user going to a lunch with her colleagues would probably rather like to learn about the most interesting technology advancements in more depth, to discuss and reason about them later on.

### 2.2.3 Presentation Aspect

With the first two aspects covered, we are left with the last one which is the presentation. In its simplest form, as it is not the main aim of our project, this could be explained by extending the same example with a tennis match and a technology lunch. When we have a large volume of news to show to the user, as is the case of sports news, we may opt for headlines with additional highlight sentence. On the other hand if there are few articles to be presented, as is the case with technology news, we may opt for headline with informational summary of the whole article. Besides the content and context the presentation can be adapted to the user herself and her preferences in general. More on this subject is covered also in [10] and an example of such adaptive content presentation method is shown in [11]. The authors of [11] propose a vehicle user interface that among other features uses haptic feedback to supplement the already busy driver's visual channel or delay phone ringing when the driver is in a complicated maneuver and thus enhance the overall safety.

These three aspects of course form more than a simple dependence model that we briefly outlined here. For example the type of content provided depends not only on situation at hand, but also on the user's implicit and explicit feedback on content that is provided over time and then used in the decision process.

These aspects however are not that important in the view of our project. In the rest of this work we thus aim at situations and actions as it was already outlined.

## 2.3 Use of Context in Information Recommendation

We can distinguish between two groups of context aware recommendation methods based on how they handle and act upon context information. The first group contains most of the context aware methods and the methods in it have a common approach of explicit application behaviour definitions for different situations which are the same for all users. Since there is no learning and intelligent or decision process involved we refer to them as to naïve context aware methods. Their main characteristic is that they do not model context explicitly but rather define application logic directly based on occurrences of particular situations. This group usually contains methods that deal with the sensor device context and adapt the content presentation to it.

The second group contains methods that use context in a more sophisticated way. These methods either combine some form of machine learning techniques or inference mechanisms that shifts them more to the intelligent groups of methods and we refer to them as to intelligent context aware methods. Their main characteristic is that they model the context explicitly and the models are thus separated from the application logic. It is also noteworthy that applications that allow users to set the context to action rules manually may be considered personalized to some extent but we still classify them with naïve methods as they lack any autonomy in doing so.

### 2.3.1 Examples of Naïve Methods

The naïve group of methods encompasses most nowadays methods that are proposed among both research and development communities. We present here a few examples together with an explanation of situation, action indicator and action concepts as we see and use them in our project.

First example from the group of naïve methods is a very simple form of context adaptation for automatic icon arrangement presented in [12]. Authors of the paper first performed a study where they let different users arrange service icons on a grid-based menu in different contexts. What they found out was that icon arrangement depended on the situation that the users were in. These results then served as inspiration for a context aware client for mobile services that is presented in the paper as a prototype. Prototype explicitly defines a set of situations and icon arrangements for them. This set of situations is mostly

based on location and contains situations of user at a market, user shopping in a mall, user at an airport and user at a cafeteria. These are all situations belonging to the location class or in this case also activity as the location is tightly bound to it.

Because the icon arrangements for recognised situations were defined explicitly by experts (users who engage in such situations most often) we classify this method as naïve. However an extension is possible where the arrangement would be learned over time by following what services are used by a particular user in a particular situation. In such case use of a service would be an indicator that in such particular situation this specific service should be moved to the main screen or some other adaptive part on the user's device. In our terms we would refer to the movement as action. The distinction between action and action indicator is that more indicators can suggest the same action in different scenarios. As it is a very simple illustration the mapping here is one to one.

Many similar examples also come from the field of adaptive mobile guides. Museum guides, navigation systems and shopping assistants as the three most frequent types of adaptive mobile guides are presented in [13]. In all the presented methods the situations are the user's current location and heading direction, and actions are again explicitly defined beforehand. All of the presented applications are aware of user's surroundings using current location of the user, her orientation and the layout plans of the place where they offer their guidance service. Museum guides use this information to find out what exhibition is the user currently looking at and for bringing detailed information about it or showing the route to a desired exhibition. The navigation systems and shopping assistants work alike with minor changes due to their domain specifics.

An approach for semi-automated context-based adaptation is presented in [14]. The paper presents Context Studio, an application where users can define rules in form of actions and their triggers, in our terminology situations. The problem of this is this kind of approach is only suitable for simplistic scenarios where the situation-action connection is straightforward. Once the inner workings become more complex such approach cannot be efficiently used or used at all. Moreover often even users are not capable of explicitly stating when the right time or situation is. The authors however present an interesting point by proposing that all situations are assigned to a class and present in an ontology that makes them more organised. This can be further on used as basis for rule extension to mitigate the sparsity problem when it often takes too long before all cases are covered. An ontology-based model for contextual content recommendation is also presented in [15] and an approach for the enhancing meaning of user's context using context ontologies is introduced in [16] and [17].

### 2.3.2 Examples of Intelligent Methods

There are tasks and problems that the naïve methods are well suited for. These are however mostly tasks that are straightforward and action to situation mapping can be done upfront. In other words naïve methods are suitable for stable and predictable domains and environments. When we however need a model to be personalized or adaptive in general we need more sophisticated methods. Examples of such methods are presented in this section.

The first method in the group of intelligent methods is presented in [18]. The authors present a method that helps users automatically switch profiles according to a sensed situation. The aim is to set the device to silent profile when the user is on a meeting or in similar situation where it would not be appropriate for the phone to ring. On the other hand when this is no longer the case a user usually wants the phone to ring in order not to miss an incoming call. To achieve the described behaviour the method exploits data from GSM base stations and nearby Bluetooth sources. The tracked data contains number of devices, their types, proximity inferred from the signal strength and device names. The situations here are different setups of device numbers, types, proximities and names, and switching to specific profiles represents actions. The important difference however that distinguishes this method from all naïve methods presented in previous section is that there is no way of knowing how to map actions to situations. We do not know the target environment and thus cannot decide in advance how many devices around at what proximity is an indicator for a specific action, in this case switching to another profile. Even though we can make some initial assumptions, the final model has to be learned over time.

Another example of an intelligent proactive application is presented in [19]. The authors present a method for recommending leisure time activities based on what time of week it is and what venues are close to the reported GPS position. While initially the presented method works with a default model, as the user chooses among proposed activities the model is changed to better reflect what the specific user really prefers to do at any time of week considering the specific options available. The situation here is composed of information about the time of week and nearby facilities and action indicators are choices the user makes at these particular situations. The indicators are then transformed into action recommendations when such situation occurs again so we can see that the three part model of situations, action indicators and actions can be applied here as well.

The last example presented in [20] brings intelligent actions based on recognized situations to the field of music. The authors present a method that takes time information as situation and played genre of music as action. The

time information contains time of day, day of week, and season and holiday information as basis for choosing what genres to play. Even though there is no classic proactivity, because we probably do not want our mobile device to start playing music at will, we still list it here because the way in which the songs are chosen is relevant to our problem at hand. The authors also introduce raw context fuzzification. They point out that if we use raw context data we face the data sparsity problem and are eventually forced to do approximations and best guesses to come with an answer. This would make the whole method much more complex and can be easily avoided by representing the raw context in concepts. For example in case of time of day we rather want to represent it in terms like morning, noon, afternoon than as hours or even minutes as it was the case in [19]. A step further is presented in [21], where the authors introduce context hierarchies even though the demonstration example that the paper provides may not be the best illustration possible. Such context hierarchies however allow for avoiding the data sparsity problem to some extent by filling the gaps with related concepts or in this case context.

As is the case with our examples, most context-aware methods exploit time and location information. We have focused our attention to examples with mobile devices, but there are also methods that use context information outside the mobile device platform. For example in [22] the authors present a method for collaborative link recommendation based on a user's location which can be used both within mobile device browser and classic non-portable computers.

## 2.4 Context-Awareness Problems

On one side context allows us to be more adaptive and at times act more intelligently. On the other side context awareness brings problems and challenges that have their root in basic characteristics always present with context [23]:

- Context information exhibits a range of temporal characteristics
- Context has many alternative representations
- Context information is imperfect
- Context information is highly interrelated

### 2.4.1 Context exhibits a range of temporal characteristics

The fact that context information exhibits a range of temporal characteristics is fairly obvious from the definition of what a context is. It is information that

describes the user's environment and situation she is in. Even though some context information is valid for a longer period of time, it is still just a definite time span and thus the temporality principle holds true. This is one of the reasons why there is a need for intelligent methods capable of learning and change over time.

### 2.4.2 Context Has Many Alternative Representations

Alternativity in context information representation relates to a problem of context acquisition and imperfection. For example we can represent location as a set of exact coordinates, as an address, facility type or as something completely else. Each of the views therefore provides substantially different sort of information, often suitable for use in different scenarios. For example while in the domain of mobile guides we are perfectly okay with exact position, in an service that finds busses and routes we may need the street information. And if we want to push the news with content related to the purpose of the facility the user is currently in, we could probably make a good use from knowing what kind of facility it is so that we know what sorts of news to push.

### 2.4.3 Context information is imperfect

Imperfection of context information is fairly obvious as well. Some context information like time can be acquired with sufficient precision, but we are talking here about context as a whole and that is much more difficult to grasp. While our minds are trained to capture every nuance of a situation we are in, the capabilities of computer devices are limited to just a few, mostly factual, information sources that cannot really capture the whole picture, but just a few details or clues in this case.

We explain in Section 2.4.2 that it is fairly easy to acquire raw coordinates of latitude, longitude and altitude from a GPS sensor, but end-user services and applications are often interested in an alternative representation. These alternative representations may be for example street and number, facility that user is in or even the storey. Such information however often has to be derived from already imperfect information.

Also, a very good example of context acquisition problem is finding out what kind of people are around. Are they close friends, family or colleagues. We do not even have to go that far and just the information whether a user is travelling with somebody or alone is not readily available. This would have to be learned over time for example by observing devices around the user. The application could after a while to a certain extent tell if the user is with

somebody or alone in terms of known people by identifying known devices in the user's proximity. Yet the information derived would still be far from perfect.

Those are just two simple examples of context information that is easy for people to sense, but acquiring them on an electronic device constitutes a great challenge. The reason why we have to address this issue of context acquisition is because we want to map situations to actions and the context that can be sensed directly may not contain information about the situation from the point of view we need.

#### 2.4.4 Context Information is Highly Interrelated

Another context information characteristic we cover in more detail relates to the problem of context selection. In other words how do we know what context information is relevant for a particular user and what is not. The first reason that is not as crucial for providing quality results is battery life. By cutting the required context information to the minimum we can substantially reduce the need for computing time and thus extend the battery life on our target devices. Second, more quality related reason, is that by using either less or more context information that is relevant to a problem at hand we are making our results worse.

The simplest example takes only two context dimensions that are used most often - time and location. Let us imagine a user that comes every day to work at seven in the morning. As it is with mornings we usually need some warm-up time so after coming to the work the user reads some news now and then. This is one example of a setup we want to support with our proactive news recommender and the question is what context information set should we use.

If we use only time to push news and the user suddenly starts working from ten because she has flexible working hours now we would fail to deliver the news at the right time. If we use both location and time to be sure we would still be in trouble because now the time is correct early in the morning but the location is only correct later on. We would still be however better off here, because the model could be accustomed to the new situation faster as we still have half of it correct. However the best choice in this case is using just the location information. Now imagine we want to push the news during the commute to and from work. In this particular example the problem can be also mitigated by using another representation of the time context – morning, however that is not the point here. The point is that while single context information may appear important it may be something else that we should found our decisions on. This goes for two pieces of context information as well. While they both appear together at all times we cannot immediately infer that



we are in the particular situation only when both are present, because one of them may for some reason depend on the other as it was the case in our previous example.

If a method that exploits context allows for model changes, meaning that it is possible to change the model once it has been learned, the problem of context selection is partially mitigated. The time we are in a gap where we cannot do anything useful only depends on how fast the new model can be learned in that particular method. Another possibility is to use some sort of weighting scheme defined explicitly for every particular domain, as the applications in such domains should know best the importance of different context information. This would be a standard step in the phase of defining a set of context information that is used for a particular context-aware application.

Apart from the two presented options there are also a few dedicated methods focused on relevant context selection. One such method presented in [24] uses a Bayesian network containing context model that is iteratively trimmed until the minimal important set of context information is identified. The authors do not work with context values directly, but use derived concepts from those values that add a required abstraction layer above them. This is mostly necessary for quantitative integral or continuous values where there are otherwise too many possibilities and no point in attempting to treat every single value as a different scenario.

## 2.5 Privacy Concerns

Using context information can at some point become a problem due to privacy concerns that may arise. For example only the location alone is a very sensitive piece of information and even though we see a clear trend of users willing to share more personal information, many of them are still conservative. This problem can be resolved using some of the anonymization methods that we present in here.

One example of applying privacy principles is presented in [25], where the described contextual framework – SocialFusion features an anonymization layer that handles any information release and ensures that this released information satisfies a K-anonymity criterion. This means that all other context information handlers work with sufficiently anonymized data and thus the user’s privacy remains protected. In short k-anonymity is a principle that goes beyond just removing data that could identify a user directly (name, social security number, etc.), because she could be just as easily identified from other data like birth date, gender, ZIP code and such. K-anonymity criterion ensures that a set of

released data is indistinguishably assignable to at least  $k$  users, while maintaining the truthfulness of the data. More on the  $k$ -anonymity principle is explained in [26].

Another approach is presented in [27], where the authors build up on hypothesis that by having background knowledge of the data, an attacker is able to compromise the  $k$ -anonymity principle and identify a user or a characteristic that an attacker is interested in, because all  $k$  users share it. The latter attack is called homogeneity attack. Besides giving analysis of the discovered attacks the paper proposes another method called  $l$ -diversity that also considers the data distribution within the identifiable group dictating that each attribute has at least  $l$  different values.

The  $l$ -diversity is further extended by [28] into an approach called  $t$ -closeness, which ensures that the distribution of any attribute values within a subset is close to the distribution of the whole set.

All of the three methods presented here have the common purpose of ensuring that a released set of information about a user cannot be used to identify her. This is a problem that has to be tackled when designing a framework for access to this information. In practice that means filtering out context information that can help in identifying a particular user. On one side this is very handy for users, because they can be sure that whatever is sent to the target recommendation framework cannot be used deceptively. On the other side this means less information for us to base our decisions on, and more importantly it can be the most important information that is missing.

Our aim is designing a method that is capable of handling different types of contextual information in various domains. With all the types of contextual information available and the alternative representations in all of them it is not possible to cover every single possible type of context. Moreover as the field of context-awareness is explored in more depth new contextual information types will be uncovered. This means that our method has to work without the background knowledge on the situational information provided. As it happens, this constraint works to our advantage considering end user privacy because whatever our action suggestion method receives cannot be used to identify the user, because there is no background information attached to it. Of course as pointed out and illustrated in [29] with the advancement in the processing power and storage capabilities everything can be performed on the client-side, possibly rendering this concern void.

# Chapter 3

## Domain of Internet News

Contextual news access aims at providing articles relevant to the user's current context. This can be for example currently viewed web page where its content model is retrieved and relevant articles about the discovered topic are recommended. Or it can be a service that recognizes what the user is currently doing and pushes news like movie inside stories or sports news accordingly. Contextual news access belongs among four types of adaptive news access methods [30]:

- News Content Personalization
- Adaptive News Navigation
- Contextual News Access
- News Aggregation

We have chosen the internet news to be our first target domain because it offers possibilities to test most aspects of our designed method. Specifically we opted for a proactive news service that is able recognize when it is the right time to push some news to the user's device. In other words we are looking for situations that are suitable for an action of pushing news. A reason for this choice is also the fact that there is very little work done on this problem. However to better understand the domain and its merits we devote this chapter to studies about user behaviour and introduction of some existing methods and applications in the internet news domain.

### 3.1 User Behaviour Observations

#### 3.1.1 Google News Experiment

In [31] the authors present an experiment using one year data from the Google News aggregation service. They reduced this dataset to contain data of approximately 17 000 registered users and performed a set of experiments with them. At the time Google News was already a news aggregation service

containing multiple topic sections as well as a number of different web parts on the main page, one of which was a collaborative based story<sup>2</sup> recommender,

The result of their experiment showed that preferences of every user consist of two parts – genuine and trend preferences. Genuine preferences are derived from the user’s profession and other interests while trend preferences depend on what is generally popular. The trend preferences also showed a certain level of dependency on the user’s location, meaning where a particular user lives. For example American users showed less interest in Euro Cup than European countries and on the other hand interest in sport sections dropped in America after the end of the baseball season, while this trend was not observed in European countries.

This insight creates an opportunity to exploit some more stable context information like the country that the user resides in to create initial models for new users based on models of users that already use an adaptive application for a longer period of time. This is of course more relevant for the content stage than for choosing the right time for a news push. A similar principle can be however also used with the time aspect taking models from family members, closest friends and colleagues, considering the social network data are readily available.

Details on how the authors used the acquired data and knowledge to improve the Google News story recommender are not that important as it is not related to our task at hand. In brief using a Bayesian classifier they introduced differentiation between genuine and trend interests based on click behaviour. This improved recommender was then evaluated with 10 000 users divided into two equal test and control groups. The users from control group got recommendation from the existing collaborative recommender while the test group users were served recommendations from the new hybrid recommender. The experiment results showed that the users from the test group used the recommender on average 30% more often. Interesting point is that they did not read more articles per visit than the control group however their visit count has risen on average by 14%. Both figures were gained from a 34 day experiment and have confidence of 99% according to the t-test.

The Google News study showed that by providing more relevant content we can make our users read more news. We believe that a similar principle applies when pushing news to a user’s mobile device. When we push news at a time that is suitable for reading them for a particular user we can this way maximize the probability of her actually opening the news. On the other hand if we push them at arbitrary times the user may not read the news even when the

---

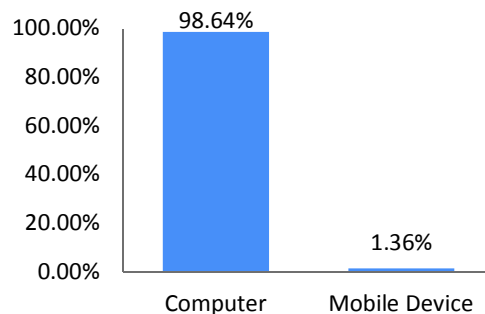
<sup>2</sup> In case of news aggregation services we refer to individual items as stories instead of articles

notification is still there when such suitable time comes and a user sees notices it.

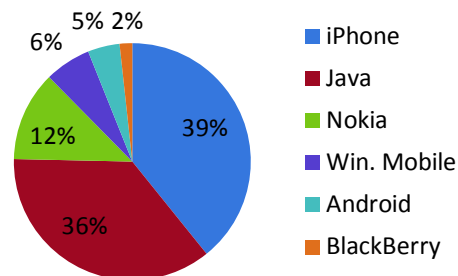
### 3.1.2 Sme.sk Experiment

Apart from external studies we have also analysed user data from sme.sk, a Slovak news portal. Our interest was particularly in distribution of visits between computer and mobile clients and patterns of accessing articles.

We have analysed approximately 136 million visits over a period of five months from the start of February 2010 to the end of June 2010 (see Figure 1 and Figure 2) and we were particularly interested in platform access distribution. The access from mobile platforms represents only 1.36%, however it still makes for over one million visits. On the other hand the analysed internet news site logs only visits to the full web site and not the mobile site. This means that there were probably more article visits from mobile devices to the mobile version of their web site which are not included in the figures. There are also mobile applications for access from different platforms that are not included in the figures as well. With all this we can see that there is already a well-established base of users who use their mobile devices for news access.



**Figure 1.** Platform Distribution.



**Figure 2.** Mobile Platform Distribution.

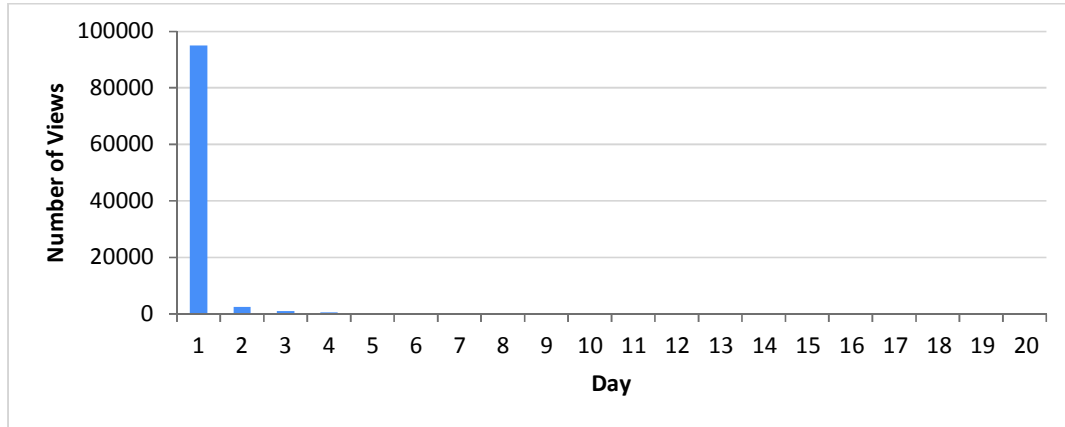
Apart from distribution of visits among different platforms we have also analysed distribution patterns of accessing individual articles and for this we have built an analysis website.<sup>3</sup> This site allows viewing hourly and daily access patterns of individual articles from the news portal as identified from the visit logs discussed earlier. The logs helped us identify three different types of articles according to the access distribution in time:

- Short-term articles
- Medium-term articles

<sup>3</sup> <http://sme.dsw.sk>

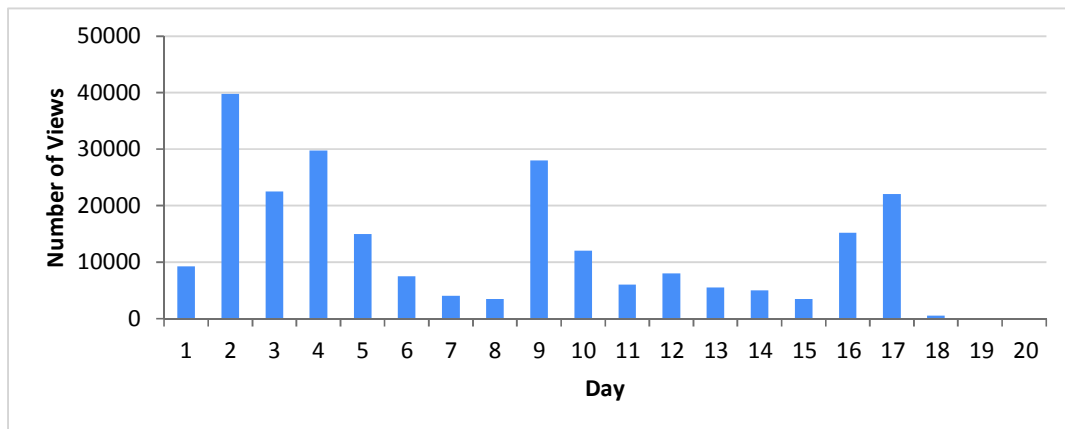
- Long-term articles

Short-term articles are articles that had most of the view counts in the first three days from the time they are published and then the interest drops to only a few visits a day. Also, in most cases of the short-term articles the most visits are during the first 24 hours. These are articles about events that happen, are discovered or announced unexpectedly. Figure 3 shows a typical distribution of a short-term article over the period of its first 20 days.



**Figure 3.** Typical view distribution of a short-term article.

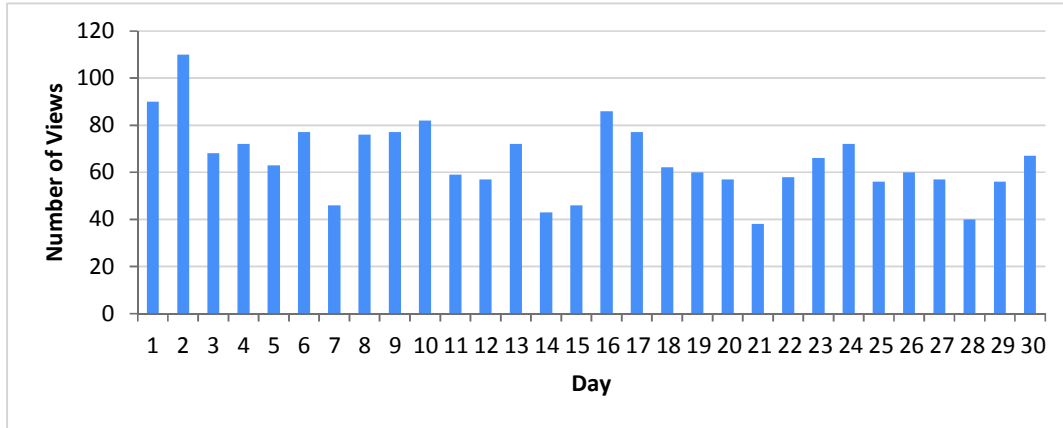
Medium-term articles show irregular spikes and gaps in different days reflecting a development of a time-spread event. These are for example articles about the World Cup schedule or an election event result predictions. Figure 4 shows a typical distribution of a medium-term article over the period of first 20 days.



**Figure 4.** Typical view distribution of a medium-term article.

Long-term articles on the other side show steady view count over a longer period of time. These are articles about time-independent topics such as yearly

horoscopes, health or cooking. Figure 5 shows typical distribution of a long-term article over the period of 30 days one month after the article was published.



**Figure 5.** Typical view distribution of a long-term article.

Because our main interest is not with the content stage we are not going to develop our own method of content recommender and instead focus on finding the right time for pushing a news article. One possibility is using an existing recommender, however we should still stick with trend articles rather than attempt to provide genuine-preference articles as failure to do so accurately may result in disinterest of reading the pushed news which may be then misleadingly interpreted as negative feedback for the current situation. On the other hand we could only in this way use trend articles that already show user interest because as it was pointed out in [32] predicting news popularity is a difficult problem and using solely textual features is not sufficient for achieving satisfying results.

The experiment results that show us three different article types considering access patterns can help us with this too. When we evaluate that it is the right time for pushing a news article to a particular user we would opt for a short-term or medium-term article only if it is in its spike. On the other hand we may opt for a long-term article in times when no option for the first two types is available. This may be during blind hours or in case the user has already read all the articles in question

## 3.2 News Recommendation

There are a lot of existing methods and applications for news access. The simplest of them most often come from the news providers directly and usually also do not provide any means of intelligent personalization. Apart from that other methods mostly focus on on-demand personalization, as it is covered in Chapter 2, but we can also see a few methods in their attempt to provide

proactive means of personalization by pushing news to a user autonomously. We provide here a few examples of both on-demand and proactive adaptive news access methods.

Article [30] covering adaptive news access in general shows such example of a proactive and adaptive news recommendation service. The presented method exploits information about currently viewed web site in the user's browser and pushes news to her accordingly. The adaptation is at the level of finding named entities like a company or actor's name that is the document's topic or just a mention within the text.

Even though the News@hand news recommendation service introduced in [33] does not involve proactive recommendation, it provides an interesting approach to news content recommendation. There are two modes built simultaneously – an interest and a context model. The interest model is built from the articles that a user accesses over time and after a stable context model is built, the interest model is linked with it. The goal is finding the right content for the right situation, which is the next step in the recommendation process after we successfully identify the right time for an article push. As presented in [34] and as we pointed out in Chapter 2 covering context in general, these basic models can be further extended using a concept ontology to provide more intelligent recommendations and overcome the data sparsity problem. A similar approach of adapting news content to weather and location context information is also presented in [35].

Most of the methods explored so far in the field of adaptive news access focus on adapting the content and do not care about other stages of the recommendation process. There are also a few methods that adapt the content presentation to the device context like screen resolution, but they are not of our primary concern. We feel that the time aspect of proactive recommendation is overlooked and time of push is generally based solely on the arrival of breaking news rather than on when it is the right time for news push to a particular user that we believe is at least as much important. Besides its straightforward goal this is also one of the reasons for our choice of first target domain application of proactive action recommendation.



## Chapter 4

# Method for Action Recommendation Based on Situation Rules

With our method we aim at providing means to support autonomous situation-based decision within end-user services or applications. We did not design our method for any specific domain, but we instead designed a method that can be used in various fields. Even though location and time context information are most widely used within context-aware methods and services we still need specific context information in the specific domains to be able to capture all the possibly important parts of the user's context. For example when designing a context-aware method for news recommendation, important contextual information may be when was the last time the user read some news. On the other hand a movie recommendation method may need to consider what the genre of the last movie was.

The aforementioned examples of domain specificity required us to design our recommendation method in such way that these specifics can be easily captured and incorporated. To achieve that we decided to build our method using symbolic representation for both user's context (situations) and actions performed by end-user services or applications that define these symbols based on what they need and consider relevant.

### 4.1 Situation Model

Situation model consists of a set of situations that describe the user's real-life situation and the situation of the environment she is in. Throughout our work we refer to types of situations (i.e. time, location, weather) as to *situation classes* and to specific situations within them (i.e. morning, home, clear) as to *situations*.

Symbolic situation representation allows for situation models that consist of relatively simple strings (symbols), but allow for representation of a variety of

situation classes, from the most simple to complex ones. Each symbol represents a particular situation and consists of two parts. The first part represents the situation class and the second part represents a particular situation within that class. An example may be situation class *Weather* with value *Clear* that would go as follows:

Weather : Clear

An instance of these situations is what is used to describe the user's context. We refer to these instances as to *situation observations*. In addition to their symbolic identification every user-situation observation has three other pieces of information:

- Time – the time at which it was observed
- Certainty – the level of certainty with the observation
- Descend rate – the speed of descending the certainty through time

The certainty value associated with it that represents how certain are we that the user is in such situation at a particular time. The values are in  $(0.0, 1.0>$  range, where 0.0 would mean that we are absolutely uncertain about the situation's occurrence whereas 1.0 means that we are absolutely sure about the situation's occurrence. This allows us to have more than one situation at a time from the same class, each with its own assigned certainty. This kind of fuzzification is very handy for symbol-based models, because it allows for representation of uncertainty as well as representation of position in a segmented continuous interval. For example at 6 AM we can say that it is morning, but at 10 AM it is starting to be lunch time, or noon as well and the concept of parallel situations with assigned certainty gives us flexibility to represent it more correctly. More on assigning certainty values to situations of different class types is covered in Section 4.1.1.

Observations are fed into our method for action recommendation at arbitrary times that depend mostly on the target client service or application and the conditions like internet connectivity, service availability or battery state. Because an observation represents a situation occurrence at a particular time, the certainty attached to the observation should not be the same sometime later (considering another situation update of the same class has not been delivered yet, which would nullify the previous one). That is why we embedded a concept of *time sensitivity* that decreases the certainty of observations as the time passes and it does it using the following formula:

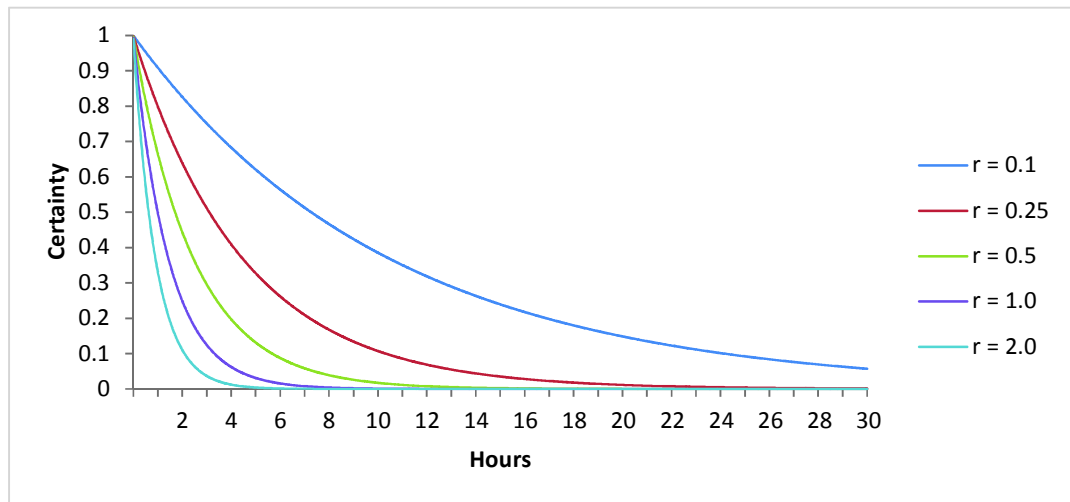
$$CF_t = \frac{CF_b}{(1 + r)^t}$$

With our method we aim at providing means to support autonomous situation-based decision within end-user services or applications. We did not design our method for any specific domain, but we instead designed a method that can be used in various fields. Even though location and time context information are most widely used within context-aware methods and services we still need specific context information in the specific domains to be able to capture all the possibly important parts of the user's context. For example when designing a context-aware method for news recommendation, important contextual information may be when was the last time the user read some news. On the other hand a movie recommendation method may need to consider what the genre of the last movie was.

The aforementioned examples of domain specificity required us to design our recommendation method in such way that these specifics can be easily captured and incorporated. To achieve that we decided to build our method using symbolic representation for both user's context (situations) and actions performed by end-user services or applications that define these symbols based on what they need and consider relevant.

## 1.6 Situation Model

Situation model consists of a set of situations that describe the user's real-life situation and the situation of the environment she is in. Throughout our work where  $CF_t$  is the resulting certainty factor at a time  $t$ ,  $CF_b$  is the base certainty initially assigned to a situation,  $r$  is the descend rate that controls how fast is the situation update losing its certainty (see Figure 6) and  $t$  is the time in hours that has elapsed since the time of the observation.



**Figure 6.** Influence of descend rate on certainty value throughout time.

The value of the descend rate for any situation observation is set by the end-user service or application as is the case with certainty values. This gives flexibility to have observations that retain their certainty over a longer period of time as well as observations that lose their certainty very fast.

#### 4.1.1 Certainty Values

Every user-situation observation has to have a certainty value associated with it. These certainty values in their basic form represent how certain the client service or application is of a situation occurrence. For example in case of an imaginary situation representing that someone is around based on Bluetooth signals the certainty may be assigned based on the strength of the signals received. Now capturing such contextual information reliably probably requires more insight and advanced approach, but we introduce it just as an illustrative example.

#### Continuous Interval Classes

Some situation classes like *TimeOfDay* divide a rather continuous interval into a set of clusters. In such case we can use the certainty not only to represent position within the situation (like for example the *TimeOfDay.Morning* would be 0.25 at 7:00, 1.0 at 8:00 and 0.25 again at 9:00) but also proximity to other situations which better models the real situation. In our case we use the normal distribution as follows:

$$CF_s = \left( \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{-\frac{(x-\mu)^2}{2 \cdot \sigma^2}} \right) \cdot (\sigma \cdot \sqrt{2 \cdot \pi})$$

where  $\left( \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{-\frac{(x-\mu)^2}{2 \cdot \sigma^2}} \right)$  is the parameterized normal distribution and  $(\sigma \cdot \sqrt{2 \cdot \pi})$

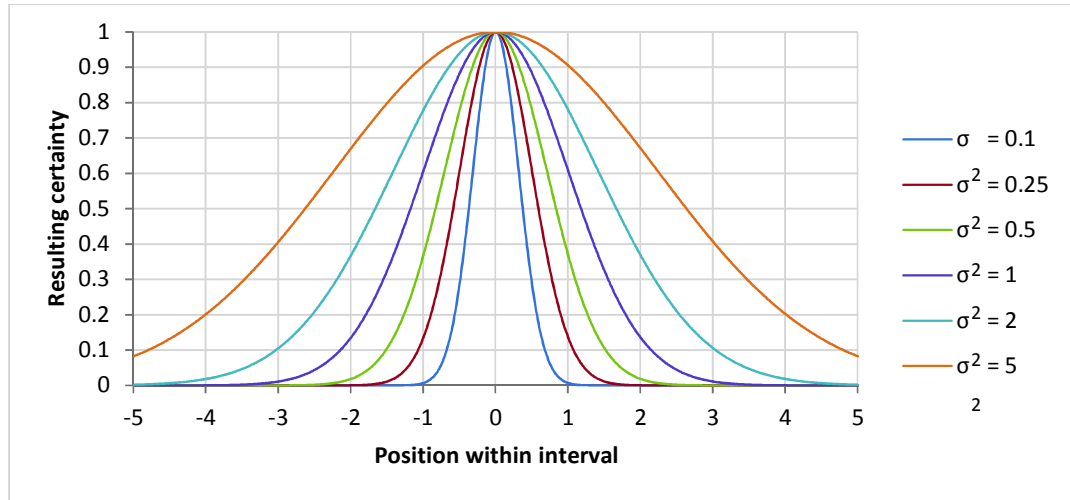
spreads the resulting certainty to  $(0,1)$  interval. The situation's original interval is transformed to  $(-1,1)$  with the value  $x$  also being transformed accordingly. The centre of an interval or mean ( $\mu$ ) is therefore represented by value 0 and values that are outside an interval are either greater than 1 or less than -1. The  $\sigma$  (variance) parameter controls how far outside intervals are certainty values still significant (see Figure 7).

With our method we aim at providing means to support autonomous situation-based decision within end-user services or applications. We did not design our method for any specific domain, but we instead designed a method that can be used in various fields. Even though location and time context information are most widely used within context-aware methods and services we still need specific context information in the specific domains to be able to capture all the possibly important parts of the user's context. For example when designing a context-aware method for news recommendation, important contextual information may be when was the last time the user read some news. On the other hand a movie recommendation method may need to consider what the genre of the last movie was.

The aforementioned examples of domain specificity required us to design our recommendation method in such way that these specifics can be easily captured and incorporated. To achieve that we decided to build our method using symbolic representation for both user's context (situations) and actions performed by end-user services or applications that define these symbols based on what they need and consider relevant.

## 1.6 Situation Model

Situation model consists of a set of situations that describe the user's real-life situation and the situation of the environment she is in. Throughout our work



**Figure 7.** Change of distribution according to  $\sigma$ .

For example in case of the aforementioned *TimeOfDay.Morning* being from 7:00 to 9:00, the value of 7:00 would be represented by -1, while the value 9:00 corresponds to 1. Going on a value of 9:30 corresponds to 1.5, which when used

within our normal distribution certainty value function with  $\sigma^2 = 0.5$  gives the resulting certainty of approximately 0.11.

Note that when using the normal distribution as we do the certainty value actually does not reach zero no matter how far from the mean we are but rather gets infinitely small. To avoid this we use a threshold value below which we consider the situation's certainty to be zero and thus not observed.

### Spatial Certainty

Some situation classes do not work with one-dimensional intervals, but map situations within two, three or even multidimensional spaces. Examples of such situation classes can be found in representing user's location. In case the situations are enclosed areas or spaces and current position is represented by a single absolute vector the certainty can be assigned similarly to how it is in case of continuous interval classes. The distance could be represented as distance of position from the centre of the shape's mass.

Often however we have the current position represented with some accuracy. That basically means that the actual position can be elsewhere within the range represented by the given accuracy. In this case we can use an area overlap model where we compute how much the two shapes overlap. Specifically in case of determining whether a user is at a certain location (i.e. *Location.Home*) we can represent both locations and current position as circles and the certainty is therefore computed as follows:

$$CF_s = \frac{\left(\frac{1}{d}\right) \cdot \sqrt{4 \cdot d^2 \cdot r_s^2 - (d^2 - r_p^2 + r_s^2)^2}}{\min(\pi \cdot r_s^2, \pi \cdot r_p^2)}$$

where  $d$  is distance between the two circle centres,  $r_s$  is radius of the situation,  $r_p$  is radius of the current position and  $r_s + r_p < d$ . This approach is suitable for computing certainties in two-dimensional spaces where both situations and positions can be represented with circles. The concept can also be scaled up and used within three or more dimensional spaces where the circles would become spheres or hyper spheres respectively.

## 4.2 Action Model

The action model consists of a set of rules that are automatically generated based on the feedback from the end-user service or application and further on the action indicators they identify. An action indicator in any specific domain is anything performed by a user that indicates an appropriate time for a particular

With our method we aim at providing means to support autonomous situation-based decision within end-user services or applications. We did not design our method for any specific domain, but we instead designed a method that can be used in various fields. Even though location and time context information are most widely used within context-aware methods and services we still need specific context information in the specific domains to be able to capture all the possibly important parts of the user's context. For example when designing a context-aware method for news recommendation, important contextual information may be when was the last time the user read some news. On the other hand a movie recommendation method may need to consider what the genre of the last movie was.

The aforementioned examples of domain specificity required us to design our recommendation method in such way that these specifics can be easily captured and incorporated. To achieve that we decided to build our method using symbolic representation for both user's context (situations) and actions performed by end-user services or applications that define these symbols based on what they need and consider relevant.

## 1.6 Situation Model

Situation model consists of a set of situations that describe the user's real-life situation and the situation of the environment she is in. Throughout our work action. For example in the domain of news recommendation a strong action indicator for recommending some news is a user actually reading some news. Another indicator may be a user playing a game on her mobile device, because it may indicate that she is possibly free to be presented some interesting news. An action of dismissing a push news notification on the other side can be indicator for not pushing any news. In case of autonomous sound profile switcher an action indicator for turning the sound off is a user actually turning it off.

The actions as well as their indicators are specific to every domain and thus as it is the case with situations, possible actions are represented using symbols and are defined by end-user services or applications. When an end-user service or application observes one of the possible action indicators, it feeds this observation in very much the same way as it does it with situation observations. The action indicator observations fed into our method serve as basis for defining rules (see Section 4.3). The set of rules that are created in this way for a particular user form her action model.

Every rule consists of two parts. The first part is a set of antecedents that define in which situation the particular rule applies. Every antecedent is defined by symbol and a certainty value. The symbol corresponds to a

particular situation from the situation model and the certainty value represents the certainty of the observation at the time when the rule was created.

Second part of every rule is its consequence or action that the rule suggests using action suggestion symbols (i.e. *ShowNews*). An action suggestion symbol is a symbol similar to the situation symbols (see Section 4.1) with the only difference being that action symbols are not defined in classes.

Every rule is also assigned a certainty value that defines what weight a rule holds. For example, considering a news recommendation service a user opening a news application is a strong clue that this may be a right situation for presenting news in future, while a user not responding to a notification is only a weak clue of the opposite. In such case for the positive feedback we would define a rule with conclusion like *ShowNews* and with higher base certainty value while in case of the weak negative feedback we would define a rule with conclusion like *DontShowNews*, but with a lower base certainty value.

The rules similarly to the situations employ a time sensitivity principle where the rules lose their certainty over time. We cannot however proceed as simply as with the situations because in case of sparse updates we would only have a couple of eligible rules to base our calculation on. Rules use the same formula for reduction of their certainty as situation with a slight modification that considers the rate at which situations from their antecedent set have appeared since the rule was defined. The final formula goes as follows:

$$CF_t = \frac{CF_b}{(1 + r)^t}$$

where  $r$  as a rate of cease is no longer a simple number present with any rule, but it is derived from the rate at which situations from the rule antecedents set have appeared since the rule was defined. The formula for finding  $r$  goes as follows:

$$r = r_b \cdot \left( \left( \frac{\sum CF_{sa_1}}{m \cdot CF_{a_1}} \right) \cdot \left( \frac{\sum CF_{sa_2}}{m \cdot CF_{a_2}} \right) \cdot \dots \cdot \left( \frac{\sum CF_{sa_n}}{m \cdot CF_{a_n}} \right) \right)$$

where  $r_b$  is the base cease rate associated with the rule (this is similar to situations),  $CF_{a_n}$  is certainty of the n-th antecedent in the particular rule,  $m$  is the base antecedent certainty cease multiplier that roughly defines how many situation updates it takes to completely cease the rule's certainty and  $\sum CF_{sa_n}$  is the sum of certainties of situation updates that match the situation of the n-th antecedent. The parameters that control the resulting cease rate are  $r_b$  and  $m$ . Their best values may differ by domain and are best to be adjusted experimentally.



With our method we aim at providing means to support autonomous situation-based decision within end-user services or applications. We did not design our method for any specific domain, but we instead designed a method that can be used in various fields. Even though location and time context information are most widely used within context-aware methods and services we still need specific context information in the specific domains to be able to capture all the possibly important parts of the user's context. For example when designing a context-aware method for news recommendation, important contextual information may be when was the last time the user read some news. On the other hand a movie recommendation method may need to consider what the genre of the last movie was.

The aforementioned examples of domain specificity required us to design our recommendation method in such way that these specifics can be easily captured and incorporated. To achieve that we decided to build our method using symbolic representation for both user's context (situations) and actions performed by end-user services or applications that define these symbols based on what they need and consider relevant.

## 1.6 Situation Model

Situation model consists of a set of situations that describe the user's real-life situation and the situation of the environment she is in. Throughout our work An intrinsic advantage of symbolic representation is that the symbols bear no meaning with them thus eliminating privacy concerns that we might otherwise experience with some users. This allows the recommendation engine to be easily deployed within any web service and extended with collaborative models, where rules can be transferred among users based on their similarity to further speed-up the training process and mitigate the cold-start problem.

## 4.3 Rule Generation

We introduced our grasp on the concept of rules in Section 4.2. We also introduced that the rules representing user's action model are defined upon a client service or application notifies of an action indicator observation. In such case the most recent situation update of all situation classes is taken to form antecedents for the newly created rules and the suggested action will be their conclusion. As we already mentioned there can be more than one valid situation from any situation class. In such case more than one rule is created in a way that every rule has exactly one antecedent from every situation class and there are rules for any possible combination of situations. Following example

demonstrates what rules would be defined in an imaginary scenario where the state would consist of five situations from three different classes:

$$\begin{aligned}
 \text{Situation} &\rightarrow C1: S1, C1: S2, C2: S1, C2: S2, C3: S1 \\
 \text{Rules} &\rightarrow \begin{aligned}
 &\text{if } C1: S1 \text{ and } C2: S1 \text{ and } C3: S1 \text{ then } A \\
 &\text{if } C1: S2 \text{ and } C2: S1 \text{ and } C3: S1 \text{ then } A \\
 &\text{if } C1: S1 \text{ and } C2: S2 \text{ and } C3: S1 \text{ then } A \\
 &\text{if } C1: S2 \text{ and } C2: S2 \text{ and } C3: S1 \text{ then } A
 \end{aligned}
 \end{aligned}$$

where the base suggestion certainty provided by the target service is distributed among the new rules based on certainty of their antecedents compared to the sum of antecedent certainty from all newly defined rules:

$$CF_r = CF_b \left( \frac{\sum CF_{a_r}}{\sum CF_a} \right)$$

where  $CF_r$  is the final rule certainty,  $CF_b$  is base certainty assigned by the client service or application,  $\sum CF_{a_r}$  is the sum of antecedent certainties from the particular rule and  $\sum CF_a$  is the sum of antecedent certainties from all newly created rules.

After all new potential rules are defined they have to be included in the user's rule base. All rules that have no existing match (their antecedent set and conclusion do not match any of the existing rules) are included in the rule base as they were defined. If a rule however matches one of the old rules, the old rule is dismissed and the certainty of the newly defined rule is adjusted as follows:

$$CF_a = \max \left( CF_{ot}, CF_r + (1 - CF_r) \cdot \left( CF_{ot} \cdot \left( \frac{CF_{ob} - CF_{ot}}{CF_{ob}} \right)^r \right) \right)$$

where  $CF_a$  is the adjusted certainty,  $CF_r$  is the new rule's original certainty computed in the previous step,  $CF_{ob}$  is the base certainty of the old rule and  $CF_{ot}$  is the certainty of the old rule at the time the new one is being defined. What the adjustment formula essentially does is that it raises the new rule's certainty by a portion of the old rule's certainty based on how much did the old rule lose its certainty over time. Because the certainty loss is based on how much were the situations from its antecedent set observed this can be interpreted as that the old rule's certainty is adjusted according to how it stood the test of time. In other words in case the same rule is defined right after its predecessor it would not gain any additional certainty from the old one. On the other side the longer the rule is in the more it suggests that it has stood the test of time and accordingly a larger portion of the remaining certainty is transferred.

With our method we aim at providing means to support autonomous situation-based decision within end-user services or applications. We did not design our method for any specific domain, but we instead designed a method that can be used in various fields. Even though location and time context information are most widely used within context-aware methods and services we still need specific context information in the specific domains to be able to capture all the possibly important parts of the user's context. For example when designing a context-aware method for news recommendation, important contextual information may be when was the last time the user read some news. On the other hand a movie recommendation method may need to consider what the genre of the last movie was.

The aforementioned examples of domain specificity required us to design our recommendation method in such way that these specifics can be easily captured and incorporated. To achieve that we decided to build our method using symbolic representation for both user's context (situations) and actions performed by end-user services or applications that define these symbols based on what they need and consider relevant.

## 1.6 Situation Model

Situation model consists of a set of situations that describe the user's real-life situation and the situation of the environment she is in. Throughout our work

The formula also contains the rate of change parameter ( $r$ ) and maximum function bound to  $CF_{ot}$ . The rate of change controls the shape of the portioning curve, where values below 1 make it shift more rapidly in the beginning while values above make for more shallow change in the beginning and swifter ones towards the end. The maximum function ensures that the certainty is not lowered in case the new rule's certainty combined with the portion of the old one's is still lower than the whole remaining certainty of the old rule. This whole model ensures continuous aggregation of rule certainties if these appear steadily over time helping us distinguish between true preferences and a possible present randomness.

## 4.4 Recommendation Calculation

The rules that are defined over time serve us as basis within the actual action recommendation process that consists of three steps:

1. Compute the importance of situation classes
2. Modify rule certainties

3. Compute the final recommendations

4.4.1 Computing the importance of situation classes

Not all of the situation classes are important for a particular user and therefore their full inclusion in the computation process degrades recommendation results. To avoid this we have to identify how important a situation class is for a particular user. To achieve this we compare how much is situation distribution from any particular class similar to a possible uniform distribution. As a result the closer the situations from any particular situation class model uniform distribution among active rules the less important the situation class is considered to be.

The first part is calculating the possible uniform distribution ratios for situations in situation classes (situation class ratio). The ratio simply reflects the representation of any given situation from a particular situation class should they be distributed uniformly. The formula for calculating situation class ratios goes as follows:

$$r_{sc} = \frac{1}{|SC|}$$

where  $r_{sc}$  is the situation class ratio and  $|SC|$  is cardinality of the situation class.

The second part is calculating the real representation of individual situations (situation ratios) within user's rules by comparing their occurrence to the occurrence of all situations from the particular class. The occurrence in this case is sum of certainties from rule antecedents and the formula goes as follows:

$$r_{sm} = \frac{\sum CF_{sm}}{\sum CF_{s_1} + \sum CF_{s_2} + \dots + \sum CF_{s_n}}$$

where  $r_{sm}$  is the computed situation ratio for situation  $m$ ,  $\sum CF_{sm}$  is the sum of certainties of rule antecedents with situation  $m$  and  $\sum CF_{s_1} + \sum CF_{s_2} + \dots + \sum CF_{s_n}$  is the sum of certainties of rule antecedents with any situation from the situation class that is being processed (the class of situation  $m$ ).

The third part is calculating the distance of each situation ratio from its situation class ratio as follows:

$$d_{sm} = |r_{sc} - r_{sm}|$$

where  $d_{sm}$  is the computed distance and  $r_{sc}$ ,  $r_{sm}$  are results from the previous computations.

With our method we aim at providing means to support autonomous situation-based decision within end-user services or applications. We did not design our method for any specific domain, but we instead designed a method that can be used in various fields. Even though location and time context information are most widely used within context-aware methods and services we still need specific context information in the specific domains to be able to capture all the possibly important parts of the user's context. For example when designing a context-aware method for news recommendation, important contextual information may be when was the last time the user read some news. On the other hand a movie recommendation method may need to consider what the genre of the last movie was.

The aforementioned examples of domain specificity required us to design our recommendation method in such way that these specifics can be easily captured and incorporated. To achieve that we decided to build our method using symbolic representation for both user's context (situations) and actions performed by end-user services or applications that define these symbols based on what they need and consider relevant.

## 1.6 Situation Model

Situation model consists of a set of situations that describe the user's real-life situation and the situation of the environment she is in. Throughout our work

The fourth and last part is computing the final importance of a situation class by comparing the average distance of situations computed in the previous step to the maximum possible average distance:

$$I_{SC} = \frac{\left(\frac{d_{s_1} + d_{s_2} + \dots + d_{s_m}}{m}\right)}{\left(\frac{(1-r_{SC}) + (m-1) \cdot r_{SC}}{m}\right)}$$

where  $I_{SC}$  is the final computed situation class importance,  $\left(\frac{d_{s_1} + d_{s_2} + \dots + d_{s_m}}{m}\right)$  is the average distance of situations ratios from their class ratio and  $\left(\frac{(1-r_{SC}) + (m-1) \cdot r_{SC}}{m}\right)$  is the maximum possible average distance. The final computed values range from 0.0 to 1.0, where exact 0.0 is achieved when all the situations are present in the user's rule base equally (exact uniform distribution) and exact 1.0 is achieved when only one situation from a class is present in the whole rule base.

#### 4.4.2 Modifying rule certainties

After we have the importance of individual situation classes ready for a particular user we can carry on with modifying the rule certainties.

The first part is modifying the base rule certainty with regards to the time sensitivity principle (see Section 4.3). This gives us rules that have their base certainty lowered based on how much the situations from their antecedents have appeared since they were defined.

The second part is computing how much each rule antecedent influences the final rule certainty based on the situation class importance and match between the rule antecedent and the current situation. The formula for computing antecedent influence goes as follows:

$$L_a = L_{a_B} + (1 - L_{a_B}) \cdot (1 - I_{SC})^{p-(p-1) \cdot L_{a_B}^{\frac{1}{q}}}$$

$$L_{a_B} = \min\left(1, \frac{CF_s}{CF_a}\right)$$

where  $L_a$  is the final influence factor of an antecedent  $a$ ,  $L_{a_B}$  is its base influence factor of antecedent  $a$  that represents the match between antecedent certainty and certainty of the corresponding situation in the current situation model ( $CF_a$  is the antecedent's  $a$  certainty and  $CF_s$  is the certainty of current situation corresponding to the situation of antecedent  $a$ ),  $I_{SC}$  is the computed situation class importance for the situation of the antecedent  $a$  and the parameters  $p$  and  $q$  ( $p, q \geq 1$ ) control the shape and the change in the shape of the curve that transforms the base influence according to the situation class importance. The resulting influence factor ranges from 0 to 1 with 0.0 being achieved when:

- Importance of the situation class is 1.0 (most important) and the situation from the antecedent  $a$  is not present in the current situation.

On the other hand a factor of value 1.0 is achieved when:

- Situation class has 0.0 importance
- There is an exact or higher match between the antecedent's certainty and certainty of the corresponding situation in the user's current situation.

Because it may be tricky to grasp the inner workings of the above formulae we present graphs (see Figure 8) that show how different base influence factors change with different situation class importance. The main idea is that if a situation from the antecedent set is not present within the current situation model the final influence factor gains significant values only when the situation

With our method we aim at providing means to support autonomous situation-based decision within end-user services or applications. We did not design our method for any specific domain, but we instead designed a method that can be used in various fields. Even though location and time context information are most widely used within context-aware methods and services we still need specific context information in the specific domains to be able to capture all the possibly important parts of the user's context. For example when designing a context-aware method for news recommendation, important contextual information may be when was the last time the user read some news. On the other hand a movie recommendation method may need to consider what the genre of the last movie was.

The aforementioned examples of domain specificity required us to design our recommendation method in such way that these specifics can be easily captured and incorporated. To achieve that we decided to build our method using symbolic representation for both user's context (situations) and actions performed by end-user services or applications that define these symbols based on what they need and consider relevant.

## 1.6 Situation Model

Situation model consists of a set of situations that describe the user's real-life situation and the situation of the environment she is in. Throughout our work class importance closes to zero. This way if an important situation is missing the rule does not retain as much certainty as it would normally do. For this illustration we have chosen  $p = 4$  and  $q = 2$ .

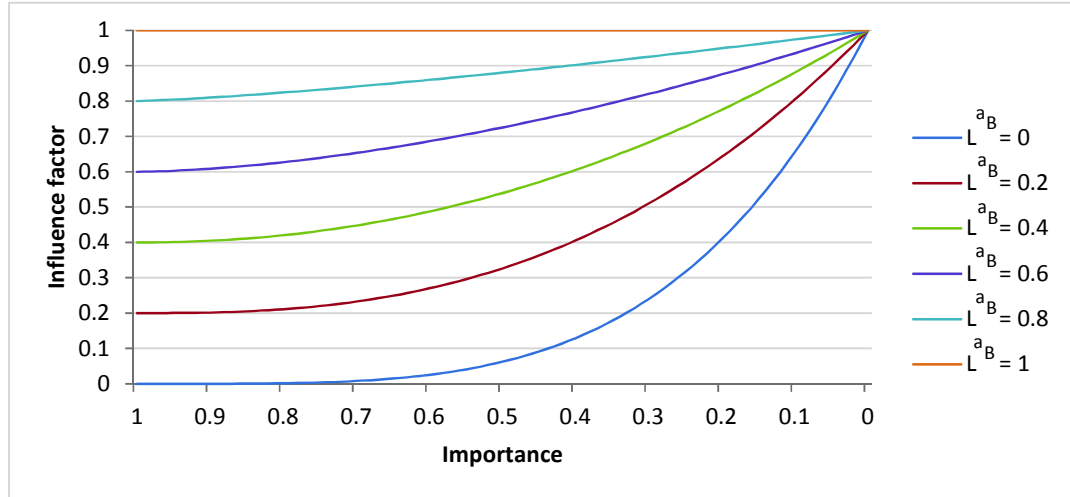


Figure 8. Influence factors and importance.

The third part is factoring the rule antecedent factors into the final rule certainty which goes as follows:

$$CF_f = (F_{a_1} \cdot F_{a_2} \cdot \dots \cdot F_{a_n}) \cdot CF_r$$

where  $CF_f$  is the final rule certainty,  $F_{a_1}, F_{a_2}, F_{a_n}$  are the factors of antecedents 1 to  $n$  and  $CF_r$  is the original rule certainty.

This model in its whole ensures that the more important are the situations that are present within the rule antecedent set, but missing from the current situation, the more is the rule certainty decreased. On the other hand if a situation with low importance is missing, the rule certainty is decreased just slightly.

#### 4.4.3 Computing the final recommendations

Computing the final recommendations groups rules from the user's rule base with matching conclusion (action) and sums their final certainties according to the summation formula proposed by David McAllister's model of working with certainties:

$$CF = CF_a + CF_b(1 - CF_a)$$

where  $CF_a$  and  $CF_b$  are two positive certainty factors that range from 0.0 to 1.0. If the conditions are held, the formula ensures that the final computed value is also within the range from 0.0 to 1.0 by adding the second certainty factor reduced by the remaining certainty. The parameters in the equation are interchangeable. Also if there are more than two certainties to sum together the order in which they are processed is not important, in other words any order gives the same result.

### 4.5 Method Characteristics and Discussion

We designed our method in a way that it can be used in any domain as both situation and action models are domain independent. One important concept in our method is the *time aspect* which allows for a change of the user's action model once her preferences change. The rate at which our method is able to change the model depends on the value of the base antecedent certainty cease multiplier introduced in Section 4.3. The lower the parameter is, the faster the model can change. However too low values can cause an adverse loss of rules even when there is no need for the model to change. Thus the most suitable



With our method we aim at providing means to support autonomous situation-based decision within end-user services or applications. We did not design our method for any specific domain, but we instead designed a method that can be used in various fields. Even though location and time context information are most widely used within context-aware methods and services we still need specific context information in the specific domains to be able to capture all the possibly important parts of the user's context. For example when designing a context-aware method for news recommendation, important contextual information may be when was the last time the user read some news. On the other hand a movie recommendation method may need to consider what the genre of the last movie was.

The aforementioned examples of domain specificity required us to design our recommendation method in such way that these specifics can be easily captured and incorporated. To achieve that we decided to build our method using symbolic representation for both user's context (situations) and actions performed by end-user services or applications that define these symbols based on what they need and consider relevant.

## 1.6 Situation Model

Situation model consists of a set of situations that describe the user's real-life situation and the situation of the environment she is in. Throughout our work value for this parameter in any particular domain is best to be found and set experimentally.

### 4.5.1 Handling the information flood

One important problem when working with loads of situation data or context is selection of the important context. Generally when automated methods work with much more information (be it context or anything else) than necessary, they tend to work incorrectly, because there are too many variables that do not matter and thus obscure what is important and invalidate the results. Our method performs well in this case of information flood. To demonstrate it, consider the following rule base:

*if C1:S1 and C2:S1 and C3:S1 ...CN:S1 then A*  
*if C1:S1 and C2:S2 and C3:S2 ...CN:S2 then A*  
*if C1:S1 and C2:S3 and C3:S3 ...CN:S3 then A*  
*if C1:S1 and C2:S4 and C3:S4 ...CN:S4 then A*

where only *C1:S1* is the important situation and the rest of them are unimportant to the current user. If we were looking for exact rules then in case

a completely new situation only with  $C1:S1$  held would arise, we would not know what to do. Our method however considers all of the four present rules, because they all contain  $C1:S1$ . Their certainty is lowered (as explained in Section 4.4) because only one of the antecedents matches the current situation, however a spike in the certainty for recommending action A is still present and can be detected. This is further emphasized by the situation class importance that is able to identify randomness among the rule antecedents. An experiment for such case is presented in Chapter 6.

#### 4.5.2 Negative certainties

It is possible to create rules with negative certainties that would lower the final recommendation certainty for any particular action, however we advise using an opposite action as it is cleaner and gives more space for making decisions on the final action. If however negative rule certainties are used there are different summation equations to work with them:

$$CF = -(CF_a + CF_b \cdot (1 - CF_a))$$

where both  $CF_a$  and  $CF_b$  are negative certainty factors ranging from -1.0 to 0.0.

$$CF = \frac{(CF_a + CF_b)}{(1 - \text{Min}(CF_a, CF_b))}$$

where  $CF_a$  is positive certainty factor ranging from 0.0 to 1.0 and  $CF_b$  is negative certainty factor ranging from -1.0 to 0.0 or vice versa.

An important note to point out is that our recommendation method and models described in this paper are not sensitive to the level at which the rule certainties are set, but the end-user service or application just needs to be consistent and assign comparably higher certainties to feedback that is a stronger clue of an action suitability and lower certainties when observing weaker clues. Because of this there is also no specific certainty at which the end-user service or application should perform a specific action, but the best time is when the certainty reaches for a significant local maximum in its development through time or is in a significant incline.

## Chapter 5

# Method for Action Recommendation in the Domain of Internet News

One of the problems in proactive news recommendation is that most today end-user services and applications consider only the content when pushing news to a user. These services mostly push news notifications when a generally and broadly important piece of news is published with some of them also considering the user's interests. There is however little work done in trying to find the right situations to push news to a user in order to maximize the chances of her actually reading them in order to maximize her overall satisfaction. This is one of the reasons we opted for the news to be our target domain and finding the right moments to be our target problem.

### 5.1 Situations

There are a few situations that we can explicitly use to tell us to what extent the user may or may not be now interested in some news content. As these form the basis of actions we discuss them later on in this chapter. What we are now more interested in is the context information that we cannot explicitly define as being either a clue for or against pushing a news notification onto a user's device. The context information sources (situation classes) that we identified with the news domain in mind are:

- Location status
- Time of day
- Day of week
- Week of month
- Month of year
- Current weather
- Calendar events
- Alarms
- Session
- Display

The situations in the aforementioned situation classes are symbols that can exist in two categories. The first category contains global symbols that are produced

by any particular client application. This is for example symbol of a user *in motion*. The other category consists of user symbols that are created during client runtime and can therefore only be recognized by that specific client instance – thus are relevant only to a specific user that defined them. An example of such situation may be a user *in motion from workplace to home*. Note that there are many users that have their *workplace to home* situations, but none of these are from the reasoning perspective semantically equal. On the other hand two users that exhibit the basic *in motion* situation are equal considering this criterion.

### 5.1.1 Location symbols

Location is undoubtedly the most commonly used context information type among context-aware services. In our method the location status can exhibit two global states:

- *AtPlace*
- *InMotion*

*AtPlace* state means that the user is currently at a place that we do not know anything else about. Similarly the global *InMotion* state signals that the user is moving between two places and we do not know anything about either one of them. Also note that when the user moves for example within her workplace premises the client should still identify this as *AtPlace* rather than *InMotion* as this context is bound to location and not movement per se.

In addition to these two global symbols the client instance can create user symbols that more specifically identify places and movements between them. A few examples of such user symbols may be:

- *AtPlace.Home*
- *AtPlace.Work*
- *InMotion.HomeToWork*
- *InMotion.WorkToHome*

These symbols help better identify specific situations and can therefore substantially improve the recommendation results.

### 5.1.2 Time Symbols

Our list of identified context information types contains four time related aspects. The first one is *time of day* which provides twelve different symbols describing current user's situation:

- *Midnight (23:00 – 01:00)*
- *Noon (11:00 – 13:00)*

- MorningNight (01:00 – 03:00)
- EarlyMorning (03:00 – 05:00)
- Morning (05:00 – 07:00)
- LateMorning (07:00 – 09:00)
- Beforenoon (09:00 – 11:00)
- Afternoon (13:00 – 15:00)
- EarlyEvening (15:00 – 17:00)
- Evening (17:00 – 19:00)
- LateEvening (19:00 – 21:00)
- EveningNight (21:00 – 23:00)

Second time related aspect is the day of week, which provides seven different symbols tagged correspondingly with the seven different days of week.

The week of month aspect provides five different symbols tagged from *First* through *Fifth*. These identify which week in order from the month start are we currently in. More specifically, the days one through seven in a given month represent the first week, days seven through fourteen are in the second week and so on.

The last time related aspect is the month of year, which provides twelve different symbols tagged correspondingly with the twelve different months in year.

### 5.1.3 Weather Symbols

The weather symbols are present in four categories, each describing a different aspect of the current weather:

- Temperature
- Wind
- Pressure
- Weather

The temperature, wind and pressure aspects have all global symbols that map to specific value ranges. In temperature category there are symbols from *ExtremeCold* through *ExtremeHot*, in the wind category the symbols are from *Calm* through *ExtremeHurricane* and in the pressure category the symbols range from *ExtremeLow* through *ExtremeHigh*. On the other hand the weather category describes the weather conditions in general and the provided symbols are:

- Clear
- Mist
- Fog
- Showers.Drizzle
- Showers.Rain
- Showers.Snow
- Showers.Sleet
- Showers.Ice

### 5.1.4 Calendar and Alarm Symbols

The calendar symbols describe user's situation relative to the events in her calendar. The calendar symbols are present in three categories

- *Calendar.InEvent*
- *Calendar.BeforeEvent*
- *Calendar.AfterEvent*

The *Calendar.InEvent* is used for describing situation of a user that is currently within a timeframe of an event from her calendar. The *Calendar.BeforeEvent* is used to describe situation when a user has an event from her calendar upcoming shortly. The *Calendar.AfterEvent* is used to describe situation when a user has an event in her calendar past shortly. Both *Calendar.BeforeEvent* and *Calendar.AfterEvent* have six different situations that describe the proximity of an event:

- Ten (0 – 10 min)
- Twenty (10 – 20 min)
- Thirty (20 – 30 min)
- Forty (30 – 40 min)
- Fifty (40 – 50 min)
- Sixty (50 – 60 min)

The alarm symbols are represented similarly with *Alarm.AlarmRingin*g, *Alarm.BeforeAlarm* and *Alarm.AfterAlarm* situation classes, where the situations in the latter two correspond to the situations in *Calendar.BeforeEvent* and *Calendar.AfterEvent*.

### 5.1.5 Display Symbols

Display symbols describe the state of the user's device display and more specifically how long has the device been in the state. There are two categories within display symbols:

- *Display.Lit*
- *Display.Shut*

Both *Display.Lit* and *Display.Shut* are described by six different situations as was the case with alarm and calendar situations (see Section 5.1.4). The only difference is that instead of ten minute intervals the display situations are segmented in five minute intervals and thus range up to thirty minutes.

### 5.1.6 Session Symbols

Session context information is the most specific to our domain and is used to describe the sessions within the target internet news application. We have divided the session information into three different categories:

- *Session.Length*
- *Session.ElapsedTime*
- *Session.ReadCount*

All three situation classes are used to describe the last session (or in case of *Session.ElapsedTime* the time that has elapsed since the last session). The *Session.Length* describes how long the last session was and is represented by five different symbols:

- Skimpy (0 – 5 min)
- Short (5 – 20 min)
- Normal (20 – 35 min)
- Longer (35 – 50 min)
- Long (50+ min)

The *Session.ReadCount* describes how many articles the user has viewed within the last session and is represented by seven different symbols:

- None (0 articles)
- Some (0 – 5 articles)
- Few (5 – 10 articles)
- Several (10 – 15 articles)
- Pack (15 – 20 articles)
- Lot (20 – 25 articles)
- Many (25+ articles)

The *Session.Elapsed* describes how much time has elapsed since the last session and is described by six different symbols:

- FewMinutes (0 – 10 min)
- QuarterHour (10 – 20 min)
- HalfAnHour (20 – 40 min)
- Hour (40 – 80 min)
- FewHours (80 – 240 min)
- ManyHours (240+ min)

## 5.2 Actions and Indicators

The situations defined in Section 5.1 belong to a group of situations that we do not initially know how to classify in terms of their suitability to pushing news to a particular user’s device because their importance is different among users. Besides there can also be no separate situation that we can classify as suitable or not, instead we have to assess them as a group.

However there are also situations that we can explicitly classify in terms of suitability for pushing news to some extent and therefore they represent our action indicators (as defined in Section 4.2). We identified four indicators that we consider suitable in the context of a news recommendation service and three indicators that suggest the time may not be right for pushing news (see Table 1).

**Table 1.** Indicators of situation suitability.

Indicators of suitable situation	Indicators of unsuitable situation
The user is reading news in our application	The user dismisses our push notification

The user is engaged in an entertainment application	The user ignores our push notification
The user is engaged in a productivity application	The user is ignoring an incoming call
The user is in a call	

Some of these indicators are easily identified while others may require more insight. For example we know for sure when a user is reading news within our application as well as we know when the user is in a call, when she is ignoring one, when she dismisses our push notification and when she ignores it.

The problem may arise when we attempt to distinguish between entertainment and production applications. The reason we want to distinguish between them is that when a user is engaged in an entertainment application, it is a stronger indication of situation suitability than when she is engaged in a productivity application. The first and obvious solution is to keep a list of these applications, where they are already classified. This however may not be the optimal solution as the number of available applications keeps growing and thus managing such lists manually is not a viable option.

A better approach is to identify the type of application by the category it belongs to. This also allows us to extend and fine-grain our model and set suitability for every category of applications, not just entertainment and productivity.

## 5.3 Domain Characteristics and Discussion

The domain of internet news is characteristic in many different ways. For example someone who reads news on regular basis may read them in the same or similar situation on weekdays, like during their commute or just after coming back from lunch. On the other hand most people do not express the same patterns on weekends, because their activity structure throughout the day is probably very different from those on weekdays. Now our aggregative rule-based method with two opposing actions as proposed in Section 5.2 is well capable of handling such scenario. Even though the rules from weekdays still apply to weekends (with lowered certainty as outlined in Section 4.5) the negative feedback on weekend produces rules with opposing suggestions that outweigh the positive ones. This concept also works when scaled up to more than just this weekly period. For example a user who is on holiday has different location situation and therefore the model can be adjusted swiftly.

A problematic scenario however still exists. Consider that we do not use the location in case of the user who leaves on holiday or that we do not capture the time of day information in case of the user whose weekend preferences differ



from those for weekdays. In both of those simple examples the behavioral patterns (feedback) change without any change in the situation models that supplement them. In other words in such case we assume that the user's behavior has changed and we should change her action model. This is the reason why end-user services and applications should do their best to capture as much of the relevant user's context as is possible. Even if we capture most of the relevant context information there will always be specific cases that simply cannot be encompassed or even anticipated. Such temporal change in overall behavior can be for example observed during holidays or with people who work in seasonal or cyclical business. What basically happens is that there is this unknown variable that is not contained within the situation model and even though the action model is shifted in these certain periods because it reacts to the change in behavior, we still want to keep the established one because it is still relevant. In other words to be able to handle unknown variables in the user's context we need to be able to identify different sub models within a model. This aspect is however not covered yet and is one of the main subjects for future work.

Reading news is for most people a kind of side activity. In most cases it is neither required by their job nor is it a dedicated hobby. Finding the right time to push some news is therefore essentially finding the right time when a user can be distracted with something that does not necessarily relate to the matter at hand, but is interesting to the user. Domains such as social network statuses and updates, movie release information, interesting images, quotes, jokes, videos and a other content that shares the characteristic of being a side activity can be therefore treated in a much similar way and the concepts presented in this chapter apply to them as well.



## Chapter 6

# Evaluation of the Method for Action Recommendation

For evaluation on our method's performance we have implemented a simulation framework and a mobile news recommendation application. The experiment was three-staged. In the first stage we performed a series of simulations to set-up our method's parameters and ensure that the inner workings are correct and can reliably identify suitable situations for a set of simulated environments and the users within.

In the second stage we distributed our modified mobile application to 6 users to evaluate on the characteristics of situation and feedback data. Our aim here was to verify the preconditions of our simulations to see if the simulated environments and users were representative of those from the real world.

In the third stage we took the knowledge from the live experiment in stage two and designed additional simulations that modeled the real environment and user behavior even more closely to their real counterparts.

For computing the certainty of actions in all of the simulations we have used only rules that were defined at least 12 hours before the time for which the recommendations were being computed in order to avoid computing recommendations for situations based on rules that were defined exactly in those particular situations.

### 6.1 Base Simulations

The purpose of these simulations is to demonstrate basic and some advanced characteristics of our rule-based action recommendation method. To achieve this we advance from the simplest simulations and scale them up as we progress. The situation classes and situations within them correspond to the ones presented in Section 5.1.

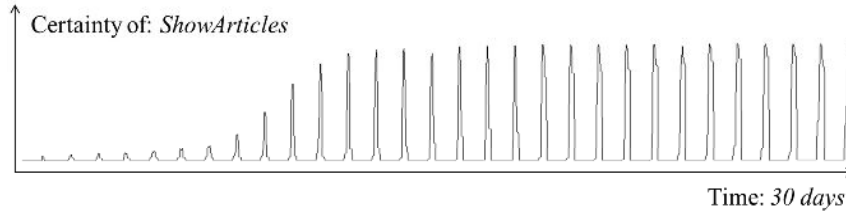
The simulation framework consists of two parts. The first part is responsible for generating an environment (situations) using situation classes, time interval and update densities as parameters. This allows us to generate scenarios with a variety of situation classes used for a variety of time spans with updates ranging from being very dense to being very sparse.

The second part is then responsible to simulate feedback based on the chosen type of user that behaves in a certain way. This can be a user that prefers to read news during mornings, a user who only reads news on showery Monday evenings or a user whose preference changes over time. These are just a few examples and the specific simulations we have performed with different environments and types of users are presented within this section.

### 6.1.1 Simplistic Scenarios

#### One Class, Simple Preference

The first simulation we ran was a simulation of environment with only one situation class and a simulation of a user whose preference was to read news in the morning from 7:00 AM to 11:00 AM (see Figure 9).



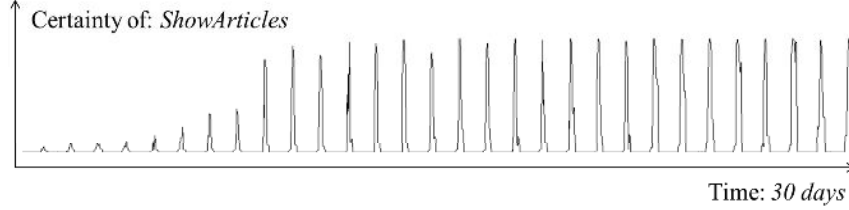
**Figure 9.** One-class, simple preference simulation results.

The environment of the first simulation contained one situation class (*TimeOfDay*) with the simulated user providing feedback in the morning which spanned over four hours, with the feedback being most probable at 9:00 AM and spreading with normal probability distribution to 7:00 AM and 11:00 AM. It is important to note however that when creating rules (providing feedback) we have always provided our method the same certainty. This corresponds with a real-world scenario where you do not adjust the feedback's certainty based on the situation's suitability because you do not know yet how suitable the situation is. The simulation ran over 30 days.

Another important point is that we have provided feedback with a certain chance, which means at random days there was no feedback provided at all. The results show clear peaks that suggest for example when it is the right time to push some news to the particular user's mobile device.

### Two Classes, Simple Preference

Our second simulation builds up on the first one with addition of one situation class with two situations (*Clear*, *Cloudy*) that represent current weather situation (see Figure 10).

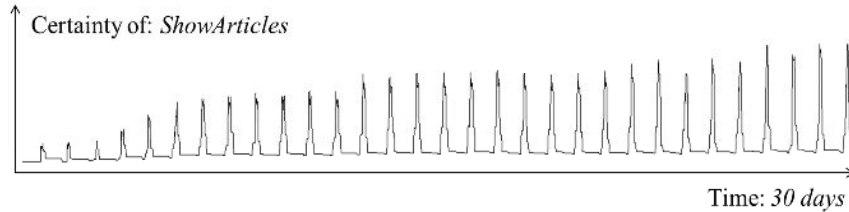


**Figure 10.** Two-class, simple preference.

The purpose of this experiment was to evaluate on how is our method capable of identifying unimportant situation classes (described in Section 4.5.1) in a very simplistic scenario. This experiment shares other characteristics with the first one. Despite a little rough start when the model was adjusting itself to the simulated user the peaks can be clearly identified both in the beginning and throughout the whole simulation.

### Two Classes, Simple Preference (Extended)

Our third simulation is a slight modification of the second simulation, where we exchanged the trivial weather situation class with class that identified days of week and thus contained seven different situations. With scaling up we can see that our method is learning a little longer than in the previous case. On the other side peaks are still clearly identifiable even within days six to ten where no feedback is provided at all (see Figure 11).

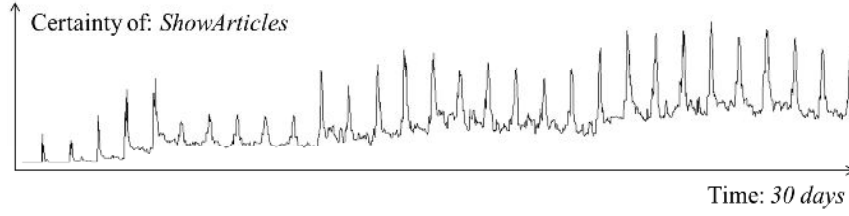


**Figure 11.** Two-class, simple preference (extended).

## 6.1.2 Complete Scenarios

### Nine Classes, Simple Preference

In our fourth experiment we scaled up number of situation classes to nine leaving other characteristics intact (see Figure 12).



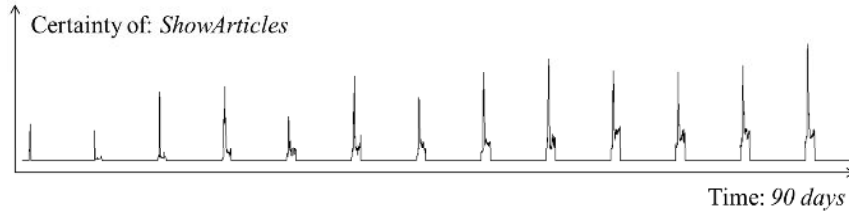
**Figure 12.** Nine-class, simple preference.

In this simulation we had eight situation classes that were unimportant for the simulated user and only one that was important. The characteristics of the eight unimportant simulated situation classes varied too. Among them were contained classes that progressed steadily with slow, average and fast rate, situation classes that changed value fairly randomly with both small and large jumps, and a situation that showed almost no change at all throughout the whole simulation.

The number and variety of unimportant situation classes employed in this simulation showed its traces in the overall appearance of the output certainty graph, but the peaks are still significant and thus reliably identifiable.

### Nine Classes, Composed Preference

In our fifth simulation we have kept all the situation classes from our previous simulation but exchanged the simulated user for one whose preference is to read news only on Monday morning. Because there are only a few Monday mornings per month we have also extended the simulation span to 90 days. The purpose of this simulation is to examine characteristics of our method when two situations from two different classes are important only when together (see Figure 13).



**Figure 13.** Nine-class, composed preference.

Because all the rules contain situation of *Monday* with 1.0 certainty it is considered to be most important. This is the reason why the certainty stays up through all Mondays. The other important class is more spread, because not only *Morning* situation is present within the rules, but there are also *EarlyMorning* and *LateMorning* antecedents within the rules. This lowers the overall class importance to somewhere around 0.7 – 0.8 and therefore even when

it is Monday evening the rule certainties do not drop to zero as the *TimeOfDay* class does not have the top 1.0 importance and the *DayOfWeek* class has.

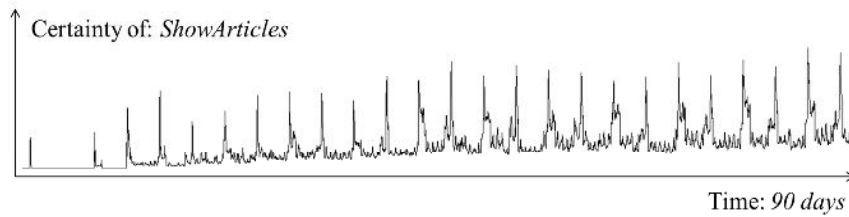
Remember that the purpose of our method is to provide significant peaks that are easily identifiable which is still held even in this borderline case. If we modified the simulated user preferences a bit making her willing to read news on Mondays both in the morning and in the evening, we would see another significant spike at the end of every Monday evening which is exactly what we need.

Besides the ability to accommodate to a user with two interlinked important situations from two different classes the results of this experiment also show our method's ability to handle rule retirement correctly. A simple time aspect model like the one used with situations would be impossible to set up with rules. It would either count with frequent feedback or with sparse feedback.

In case the model would count with frequent feedback and thus rule creation the rule certainties in this would be degraded before the next week comes. If the model would count with sparse feedback it would not allow for preference change because the rules would be held in the user's rule base for too long. As explained in section 3 our method considers the rate at which situations from rule antecedents appear over time and bases rule retirement on that.

### Nine Classes, Multiple Composed Preferences

Our sixth experiment simulation scales up the fifth one and studies a case of a simulated user for whom there are two pairs of important situations. The simulated user prefers to read news on Monday morning and Friday evening (see Figure 14).



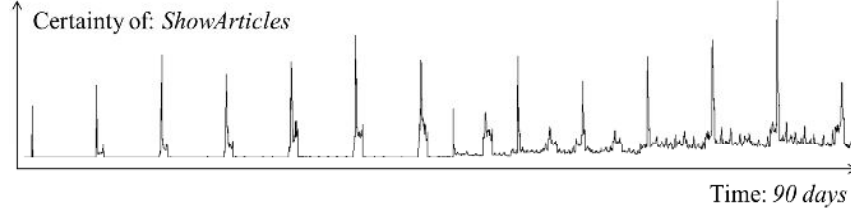
**Figure 14.** Nine-class, multiple composed preferences.

This experiment simulation was again run throughout a period of 90 days. Even though this time we have two different pairs of important situations within nine different classes our method is still able to reliably identify the important ones after only a few of their occurrences. The gap in the first week's Friday is caused by our simulated user not providing feedback on that day. There are a

couple other days where no feedback is provided, however rules from the past weeks are still in force and thus the peaks are present in the final results.

### Nine Classes, Preference Change

Our last experiment simulation tests out the rule retirement model by simulating a user who prefers to read news on Monday morning, but after one and a half month her preferences change to Friday evening (see Figure 15).



**Figure 15.** Nine-class, change of preferences.

This experiment also spans throughout 90 days and its results clearly demonstrate our method’s capabilities to change when users change their behavior. Throughout the first one and a half month the simulated user gives positive feedback on Monday morning and our method observes and learns this properly as is the case with our fifth simulation. After roughly one and a half month the user’s preferences suddenly change to Friday evening which is represented in the graph by the first spike that is closer to the previous one than the others. The spike right after for Monday morning is around the same size for it has been lowered by lack of feedback on that day, but mainly by addition of new rules for another day for the first time. This (as described in Chapter 4) lowers the importance of the *DayOfWeek* situation class. For the rest of the time allotted for the simulation the rules for Monday morning lose their certainty according to the model described in section 3 and rules for Friday evening are becoming stronger. Because the rules for Monday have little to no influence in the upcoming weeks the importance of the *DayOfWeek* situation class is raised again as well.

### 6.1.3 Experiment Summary and Discussion

The purpose of the experiment simulations was to testify that our rule-based action recommendation method and especially its mathematical models that underline it are correct. We have chosen to perform simulations with their advancing complexity to discover our method characteristics as the data complexity scales up in different directions.



The experiments have shown that our rule time sensitivity model works much more reliably than a simplistic model like the one used with situations would. In multiple scenarios we have seen that the certainty does not drop to zero even when it is clear that it is not the right situation for a particular action. This is okay, because as we already stated we are interested in significant spikes and possibly inclines of the resulting recommendation certainty.

Even though we have seen our method perform well even when scaled up, the wobbliness is present more significantly when more unimportant situation classes are employed. This is a well-known problem in recommendation where adding more inputs does not always yield better results and it is thus important to identify and use only the relevant context [15]. But again due to the underlying mathematical models of our method we are able to deal with reasonable amount of clutter very well. On the other side to make our method perform its best it is up on the target end-user service or application to define and use the classes that are important in the domain. For example in case of real push news service we would employ situation classes that tell us how long ago the last session was, how many articles the user has read during it or how long it lasted.

## 6.2 Live Experiment

The base simulations served us to set our method's parameters and ensure that the underlying mathematical model can correctly identify suitable situation within simulated environments and virtual users. To continue with our experiment and perform both environment and user feedback simulations that closely model their real life counterparts we have performed a two-week long live experiment. In this live experiment we have asked six people to install and use our simplified prototype application on their mobile devices (see Figure 16).



**Figure 16.** Live experiment application.

Their task was to regularly press one of four buttons according to how they felt about reading some interesting news or a similar activity at these times. The specific question the users answered was based on what activity that reminds reading news the user does regularly. For three of the users it was reading news, one had Facebook updates, one had tweets and the last one was answering when a good time for some fun mime drawings was.

### 6.2.1 Situation Observations

In the two-week period we have gathered over 700,000 situation observations that showed two significant characteristics:

- Gaps in situation observation
- Situation lock-in

Not all of the users had permanent internet connectivity and therefore some context information that was being gathered from web services (like weather) was not always present with situation updates. We have designed our experiment application to store context updates on the device until such time when the device got connected and then all the updates were sent at once. The situation classes that did not need internet connectivity to make their observations did not show that many gaps in observation updates even though some were still present. Because no observation updates were made at those times we expect that this was due to the users turning their devices off, batteries running out or the background process being killed by the operating system gather additional memory for a resource intensive task.

Another characteristic that can be seen in the situation observations is a kind of situation lock-in. For example because it was a very short period the temperature situations did not fluctuate that much and therefore the class was

“locked” in just a few situations for the whole time. Now because all of the rules in such environment have only a few temperature situation observations present in them once the temperature changes the certainty of recommendation for a suitable situation drops. This is however just a short-term problem because as more feedback is received the model accommodates and our method identifies the real importance of the previously locked-in situation classes.

### 6.2.2 Feedback Observations

There were more than 3,000 feedback responses from our test users and as was the case with situations we noticed two important characteristics:

- Twofold feedback
- Weak feedbacks

The live experiment took place from 2nd April 2012 and 16th April 2012 which was just around Easter and this setup manifested itself in an expected way. A few users’ feedback responses from weekdays outside the Easter holidays were quite regular, but did not correlate with the feedback responses from the same users within the Easter. This relates to the issue of discovering change in unknown context information that we already discussed in Section 5.3 as a subject of our future work.

The second characteristic we discovered is presence of weak feedbacks. By a weak feedback we understand a feedback that is just weakly supported or not supported at all by additional feedbacks. In other words the collected data contained isolated and scattered random feedbacks in situations such that they were not present in the rest of the time period.

## 6.3 Additional Simulations

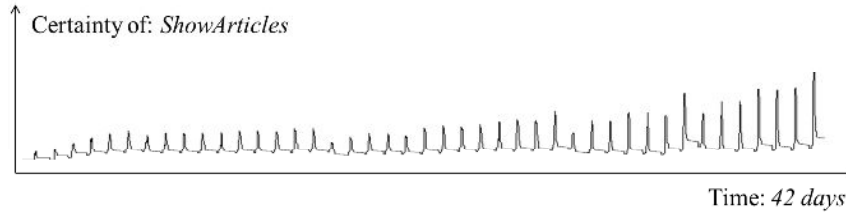
Based on the characteristics observed in the live experiment we designed a few more simulations. We mainly focus our attention on the *situation lock-in* and *weak feedback* characteristics, because the gaps in situation observation simply mean that the devices did not have connectivity to our recommendation service which would be mitigated if the engine resided right on the clients’ devices. The issue of discovering changes in unknown context variables has already been covered and is subject of our future work.

### 6.3.1 Situation Lock-in

The situation lock-in characteristic of some classes can cause a temporary drop in recommendation certainty if most of situations from a situation class are observed only after a user's model is already well-established with just a very small subset of situations from that situation class. To model this scenario we have defined an environmental simulation with the following characteristics:

- 6-week simulation
- Temperatures
  - Weeks 1 and 2: *Cold*
  - Weeks 3 and 4: *Cold*, *Fresh* and *Warm*
  - Weeks 5 and 6: *Fresh*, *Warm* and *Hot*

This setup creates a real-life like scenario where there is a steady cold temperature for a few weeks and gets warmer as the time goes on. For feedback we have chosen a user who prefers to read news every evening as this helps us visualize what is happening with the model and certainty of the recommendations. The results are shown in Figure 17.

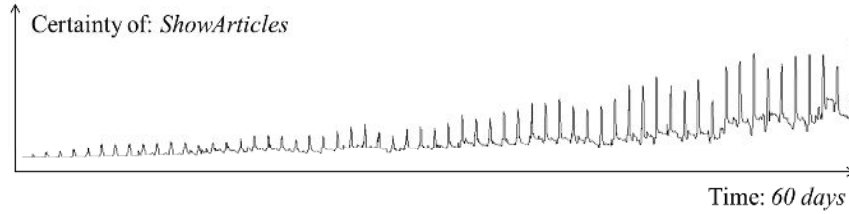


**Figure 17.** Situation lock-in simulation.

We can see that after building the user's action model for two weeks the spikes are distinct and clearly identifiable throughout the whole period. Beginning with the 14th day the temperature starts to be more volatile and even though is not important for our virtual user the recommendation spikes drop. From the beginning the spikes are again very subtle, but as the time progresses our model is adjusting itself and soon the suitable situations can be again easily distinguished. In the last part of the simulation (beginning by day 28) we add another temperature situation and retire the initial one. Slighter drop can be observed again but is not as significant as it was the first time because now the weather does not have such high importance.

### 6.3.2 Weak Feedbacks

Weak feedbacks are feedbacks from users that occur in these situations just once or very sparingly. These are feedbacks that can be from our perspective considered random because they do not depend on the captured part of user’s context, but rather on some unknown variable. The random feedback can be anywhere from sparse to very dense. In case of sparse random feedback our aim remains the same – identifying the suitable situations even with noise that cannot be handled correctly. But as the random feedback gets denser it can suggest two different things. First possibility is that there is a part of user’s context that is not captured even though it is significant in regards to the recommendation. Another possibility is that the preferences we are trying to model are simply too random. In both cases our aim stays the same and the question is how random the data can be before we can no longer reliably distinguish between spikes generated by the random feedback and the spikes that represent at least the most significant suitable situations. To evaluate on this we have performed a simulation where we included random feedback from the virtual user (see Figure 18).



**Figure 18.** 7 to 14 random feedbacks every week.

Despite a present wobbliness we can see that even when the feedback that contributes to the defined suitable situation constitutes only around one third of the total feedback the spikes in data can still be identified. This is an important fact because in the real-world use we are rarely able to capture all nuances of the user’s context that influence the real situation suitability, but we at least have to be able to capture the ones we can even in such case of considerable amount of noise.

The purpose of our three-staged experiment was to fine-tune our method and evaluate on its ability to correctly identify suitable situations for performing actions based on user’s feedback. Most of the evaluation involved our open simulation framework that allowed us to simulate both various environments and different user types. Apart from that we have also performed a live experiment to help us identify important characteristics of the real-world environment. The results of our last experiments modelling the aforementioned

### *Evaluation of the Method for Action Recommendation*

characteristics show that even in difficult scenarios, like in a situation lock-in or moderate randomness, our method is still able to provide good results.

## Chapter 7

# Conclusions and Future Work

We live in a world of pervasive computing where automated technology begins to be a part of our everyday lives. Most of the autonomous computing today however features one-size-fits-all principles where they do not distinguish the differences between individuals. A similar problem is in the domain of proactive news recommendation or interesting content provision in general where methods usually only consider the content and not for example if it is the right time to disturb the user with some or a particular type of content.

We have designed and in this work described our method for aiding with the autonomy of the pervasive services and applications by enhancing them with action recommendations based on situation rules that our methods defines from the observations and feedback received.

Evaluation of our method shows that it is capable to operate in a wide range of scenarios, from those that only require a small portion of user's context to those where a considerable amount of noise is involved or where users change their long-term behavior.

Our future work mainly involves enhancing the capabilities of our method to handle unknown variables and hidden links. This involves reducing the noise, identifying steady situation classes and characteristics of any situation class in general from the observations to better approximate their importance and solving the sub-model within model issue. Other enhancements include a support for collaborative models, where rules can be transferred and shared among similar users just as content is recommended in this collaborative way. This may significantly counter the known cold-start problem for new users, after a sufficient user-base is already active in the target service.

We have designed our method in a way that it is easily portable to any domain and the only responsibility end-user services and applications have is deciding on what actions they will be performing and what context information can be relevant to support the decision process. We believe that our method and frameworks based on it will due to their simplicity encourage more services and applications to become context-aware and more intelligent with it as well.





## References

1. Micarelli, A., Gasparetti, F., Sciarrone, F., Gauch, S.: Personalized Search on the World Wide Web. In : The Adaptive Web: Methods and Strategies of Web Personalization. Springer-Verlag, New York (2007)
2. Simpson, J., Weiner, E., eds.: The Oxford English Dictionary 2nd edn. Oxford University Press, Oxford (1989)
3. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In : Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, pp.85-90 (1994)
4. Chen, G., Kotz, D.: A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dartmouth College, Dartmouth (2000)
5. Pazzani, M., Billsus, D.: Content-Based Recommendation Systems. In : The Adaptive Web: Methods and Strategies of Web Personalization 4321. Springer-Verlag, New York (2007)
6. Schafer, B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative Filtering Recommender Systems. In : The Adaptive Web: Methods and Strategies of Web Personalization 2nd edn. 4321. Springer-Verlag, New York (2007)
7. Burke, R.: Hybrid Web Recommender Systems. In : The Adaptive Web: Methods and Strategies of Web Personalization 2nd edn. 4321. Springer-Verlag, New York (2007)
8. Pessemier, T., Deryckere, T., Martens, L.: Extending the Bayesian Classifier to a Context-Aware Recommender System for Mobile Devices. In : ICIW, St. Maarten, pp.242-247 (2010)
9. Liu, L., Lecue, F., Mehandjiev, N., Xu, L.: Using Context Similarity for Service Recommendation. In : ICSC, Pittsburgh, pp.277-284 (2010)
10. Bunt, A., Carenini, G., Conati, C.: Adaptive Content Presentation for the Web. In : The Adaptive Web: Methods and Strategies of Web Personalization. Springer-Verlag, New York (2007)
11. Bellotti, F., Gloria, A., Montanari, R., Dosio, N., Morreale, D.:

COMUNICAR: designing a multimedia, context-aware human-machine interface for cars. In : *Cognition, Technology & Work*, London, pp.36-45 (2005)

12. Böhmer, M., Bauer, G.: Exploiting the Icon Arrangement on Mobile Devices as Information Source for Context-awareness. In : *MobileHCI*, Lisboa, pp.195-198 (2010)
13. Krüger, A., Baus, J., Heckmann, D., Kruppa, M., Wasinger, R.: Adaptive Mobile Guides. In : *The Adaptive Web: Methods and Strategies of Web Personalization*. Springer-Verlag, New York (2007)
14. Korpipää, P., Häkkinä, J., Kela, J., Ronkainen, S., Käsälä, I.: Utilising Context Ontology in Mobile Device Application Personalization. In : *MUM*, Maryland, pp.133-140 (2004)
15. Zhang, Y., Cai, S., Hu, M., Liang, F.: A Study on the Method of Mobile Content Recommendation Based on Situations. In : *ISCID*, Hangzhou, pp.23-26 (2010)
16. Xiao, H., Zou, Y., Ng, J., Nigul, L.: An Approach for Context-aware Service Discovery and Recommendation. In : *ICWS*, Miami, pp.163-170 (2010)
17. Costa, A., Guizzardi, R., Guizzardi, G., Filho, J.: CORES: Context-aware, Ontology-based Recommender system for Service recommendation. In : *CAISE*, Trondheim (2007)
18. Ala-Siuru, P., Tapani, R.: Understanding and recognizing usage situations using context data available in mobile phones. In : *ubiPCMM*, California (2006)
19. Roberts, M., Ducheneaut, N., Beloge, B., Partridge, K., Price, B., Bellotti, V., Walendowski, A., Rasmussen, P.: Scalable Architecture for Context-Aware Activity-Detecting Mobile Recommendation Systems. In : *WoWMoM*, California, pp.1-6 (2008)
20. Shin, D., Lee, J.-w., Yeon, J., Lee, S.-g.: Context-Aware Recommendation by Aggregating User Context. In : *CEC*, Vienna, pp.423-430 (2009)
21. Biegel, G., Cahill, V.: A Framework for Developing Mobile, Context-aware Applications. In : *PerCom*, Orlando, pp.361-365 (2004)
22. Brunato, M., Battiti, R., Villani, A., Delai, A.: A Location-Dependent Recommender System for the Web. Technical Report DIT-02-0093, Trento University, Trento (2002)
23. Henriksen, K., Jadwiga, I., Rakotonirainy, A.: Modeling Context

- Information in Pervasive Computing Systems. In : Pervasive, London, pp.167-180 (2002)
24. Yap, G.-E., Tan, A.-H., Pang, H.-H.: Discovering and Exploiting Causal Dependencies for Robust Mobile Context-Aware Recommenders. *IEEE Transactions on Knowledge and Data* 19, 977-922 (2007)
  25. Beach, A., Gartrell, M., Xing, X., Han, R., Lv, Q., Mishra, S., Seada, K.: SocialFusion: Context-Aware Inference and Recommendation By Fusing Mobile, Sensor, and Social Data. Technical Report CU-CS-1059-09, University of Colorado, Boulder (2009)
  26. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Samarati, P.: k-Anonymity. In : *Secure Data Management in Decentralized Systems*. Springer-Verlag (2007)
  27. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramanian, M.: l-diversity. In : *ICDE*, pp.24-35 (2006)
  28. Li, N., Li, T., Venkatasubramanian, S.: Privacy Beyond k-Anonymity and l-Diversity. In : *Proceedings of IEEE International Conference on Data Engineering* (2007)
  29. Gerber, S., Fry, M., Kay, J., Kummerfeld, B.: PersonisJ: Mobile, Client-Side User Modelling. In : *UMAP, Big Island*, pp.111-122 (2010)
  30. Billsus, D., Pazzani, M.: Adaptive News Access. In : *The Adaptive Web: Methods and Strategies of Web Personalization*. Springer-Verlag, New York (2007)
  31. Liu, J., Dolan, P., Pedersen, E.: Personalized News Recommendation Based on Click Behavior. In : *IUI, Hong Kong*, pp.31-40 (2010)
  32. Krestel, R., Metha, B.: Predicting News Story Importance using Language Features. In : *WI-IAT, Sydney* (2008)
  33. Cantador, I., Castells, P.: Semantic Contextualisation in a News Recommender System. In : *Workshop on Context-Aware Recommender Systems*, New York (2009)
  34. Ijntema, W., Goossen, F., Frasincar, F., Hogenboom, F.: Ontology-Based News Recommendation. In : *EDBT/ICDT, Lausanne* (2010)
  35. Yeung, K., Yang, Y.: A Proactive Personalized Mobile News Recommendation System. In : *DeSE, London*, pp.207-212 (2010)



ODPORÚČANIE INFORMÁCIÍ S VYUŽITÍM  
KONTEXTU V ŠPECIFICKEJ DOMÉNE

(RESUMÉ V SLOVESKOM JAZYKU)



# 1 Úvod

Odporúčanie informácií je vo svojej základnej forme proces skladajúci sa z troch krokov. Prvým krokom je rozhodnutie kedy budeme odporúčať. Potom sa musíme rozhodnúť čo budeme odporúčať a v akom množstve a nakoniec ako tieto odporúčania prezentujeme používateľovi. Väčšina dnešných odporúčačov sa však dnes zameriava výhradne na časť obsahu, považujúc ostatné rozhodnutia konštantné. Toto je v poriadku, pokiaľ ide o odporúčanie „na požiadanie“, teda odporúčanie, ktoré nasleduje po určitej forme dopytu. Týmto môže byť napríklad usporadúvanie výsledkov vyhľadávania alebo odporúčanie ďalších produktov v internetovom obchode. V týchto prípadoch sa nemusíme starať o čas odporúčania, pretože niekto iný, najčastejšie používateľ, robí toto rozhodnutie sám.

Na druhej strane vo svete všadeprítomných počítačov môžeme vidieť stále viac a viac pokusov o inteligentné proaktívne odporúčače. Proaktívny odporúčač je taký, ktorý sa zaoberá prvým krokom v procese odporúčania – rozhodovaním, kedy sa odporúčanie uskutoční. Toto rozhodnutie o čase je zvyčajne založené na identifikácii situácií a ovplyvňuje obe ďalšie, pretože je medzi nimi často veľmi silná väzba. Napríklad obsah odporúčaný takýmto proaktívnym odporúčačom internetových noviniek by mal poskytnúť iný obsah pokiaľ je používateľ na športovom stretnutí so svojimi priateľmi a iný pokiaľ je napríklad v kine so svojimi blízkymi.

Naším hlavným cieľom je navrhnúť a vytvoriť metódu schopnú efektívne odporúčať akcie na základe situácie používateľa a navrhnúť túto metódu tak, aby bola jednoducho aplikovateľná v ľubovoľnej doméne a vývojom služieb a aplikácií pre koncových používateľov zahrnúť do svojich dizajnov kontext s malým či takmer žiadnym úsilím.

## 2 Proces odporúčania informácií

Vo všeobecnosti existujú z pohľadu iniciácie odporúčania dva typy odporúčačov. Prvým typom sú odporúčače „na požiadanie“ a druhým typom sú proaktívne odporúčače, na ktoré je v našej práci zameraná hlavná pozornosť. Proaktívne odporúčanie sa skladá z troch aspektov:

- Čas odporúčania

- Odporúčaný obsah
- Prezentácia odporúčaného obsahu

Všetky tri aspekty tvoria jednu ucelenú jednotku, kde každý aspekt závisí na ostatných dvoch a ignorovanie týchto závislostí môže viesť k podstatnému zníženiu kvality výsledkov odporúčania. Našu pozornosť v ďalších častiach práce zameriavame na aspekt času, ktorý je našim hlavným záujmom, no ešte predtým uvádzame koncepty *situácia* a *kontext* a pohľad na ne z perspektívy našej práce.

## 2.1 Kontext a situácie

Jedna z definícií v [1] uvádza, že situácia je pozícia osoby voči okolnostiam. Z nášho pohľadu teda okolnosti charakterizujú stav používateľa alebo prostredia, v ktorom sa nachádza, čo môže byť napríklad aké je miesto, čas či počasie tam, kde sa používateľ nachádza ale taktiež čo robí alebo kto je naokolo. Čo sa týka kontextu, tak existuje mnoho definícií o tom, čo vlastne kontext je a ako ho delíme. Z nášho pohľadu je však najzaujímavejší tzv. obsahový pohľad, pretože tento môže byť použitý na kategorizáciu kontextu podľa použitia v jednotlivých fázach procesu odporúčania. Obsahový pohľad rozlišuje tri kategórie kontextu:

- Kontext zariadenia
- Kontext používateľa
- Kontext prostredia

Kontext zariadenia zahŕňa informácie o zariadení používateľa ako napríklad úroveň nabitia batérie, rozlíšenie displeja či kvalitu pripojenia na internet a je najhodnotnejší pre metódy, ktoré sa zameriavajú najmä na aspekt prezentácie. Kontext používateľa a kontext prostredia na druhej strane opisujú situáciu, v ktorej sa používateľ nachádza a teda sú našim hlavným záujmom. Rozdiel medzi kontextom používateľa a kontextom prostredia je v tom, na čo sa viažu viac, ale pretože tento rozdiel nie je pre nás podstatný, budeme oba nazývať spoločným názvom kontext používateľa a prostredie, resp. iba kontext používateľa. Nasleduje niekoľko príkladov často používaných kontextových informácií:



- Dátum a čas
- Poloha
- Kalendár používateľa
- Galéria multimédií používateľa
- Sociálne siete
- Uložené dokumenty
- História vyhľadávania
- Počasie
- Zariadenia v blízkosti
- Ľudia v blízkosti

## 2.2 Aspekt času pri proaktívnom odporúčaní informácií

Proaktívne odporúčače musia rozpoznať kedy je správny čas na odporúčanie, pričom v najtriviálnejšom prípade môžu byť tieto časy explicitne definované tvorcom. Napríklad pre odporúčanie noviniek to môže byť pár minút po tom, ako zazvoní alarm, pár minút pred odchodom na obed alebo večer po tom ako v izbe zhasnú svetlá. Príklad takejto naivnej metódy je uvedený napríklad v [2], kde autori na základe experimentov explicitne navrhli preusporiadanie ikon na základe toho, kde sa používateľ nachádza (zahrnuli trhovisko, obchodný dom, letisko a jedáleň). Veľa ďalších príkladov naivných metód pochádza aj z domény mobilných navigátorov prezentovaných v [3] a príklad poloautomatizovaného vykonávania akcií na základe používateľmi definovaných pravidiel je uvedený napríklad aj v [4]. Problém je v tom, že takéto prístupy sú vhodné iba pre jednoduché scenáre, kde sú prepojenia situácií na akcie jednoduché a jednoznačné. Keď začnú byť však tieto vzťahy komplexnejšie, kedy ich už ani sami používatelia nevedia presne definovať, tak je použitie týchto metód nepraktické a ich možnosti obmedzené.

Na druhej strane oproti naivným metódam stoja tzv. adaptívne metódy. Príklad takejto adaptívnej metódy je uvedený napríklad aj v [5], kde autori prezentujú metódu schopnú automaticky prepínať profily telefónu podľa potrieb používateľa, a to na základe zariadení, ktoré sa nachádzajú v blízkom okolí. Ďalší príklad adaptívnej metódy je uvedený v [6], kde autori prezentujú metódu na odporúčanie aktivít vo voľnom čase, ktorá spočiatku preferuje najbližšie možnosti, no postupom času sa prispôsobí a odporúča aj na základe používateľom preferovaných typov zariadení pre špecifické situácie.

## 3 Doména internetových noviniek

### 3.1 Existujúce riešenia

V rámci internetových noviniek, ktoré sme si vybrali ako našu prvú cieľovú doménu už existuje niekoľko metód, ktoré sa zaoberajú nielen odporúčaním noviniek ako takým, ale práve kontextovým či dokonca proaktívnym odporúčaním noviniek. Prvý príklad je napríklad v [7], kde autori prezentujú svoju metódu proaktívneho odporúčania noviniek na základe aktuálne prezeranej webstránky. V [8] autori prezentujú svoju koncepciu News@hand, v ktorej síce nejde o proaktívne odporúčanie, ale kontext využívajú na rozlíšenie medzi všeobecnými záujmami používateľa o rôzne typy článkov a záujmami, ktoré vyplývajú z jej aktuálnej situácie, teda z kontextu.

### 3.2 Správanie používateľov

V [9] autori prezentujú experiment, v ktorom využili agregáčnú službu noviniek Google News na zistenie správania sa používateľov pri ich čítaní, najmä z pohľadu využitia a efektivity odporúčaní. V experimente zistili, že preferencie používateľov sa skladajú z pravých a trendových preferencií, kde pravé preferencie sú zvyčajne odvodené od profesie, či záujmov konkrétneho používateľa a trendové závisia od aktuálnych široko záberových udalostí ako napríklad olympiáda, majstrovstvá sveta, voľby a podobne. Tieto fakty zobrali autori do úvahy pri návrhu novej verzie kolaboratívneho odporúčača ktorú následne vytvorili a experimentálne nasadili. Výsledkom ich experimentu bolo, že skvalitnenie odporúčaného obsahu nielenže zvyšuje počet návštev odporúčaných článkov v pomere ku všetkým navštíveným, ale aj zvyšuje frekvenciu návštev používateľov. Na druhej strane priemerný počet prečítaných článkov počas každej návštevy ostal približne rovnaký.

Experiment pre zistenie správania sa používateľov sme vykonali aj my, no náš záujem bol zameraný na zistenie práve trendových charakteristík prístupov k článkom. V experimente nad päťmesačnými dátami z portálu internetového denníku sme.sk sme analyzovali približne 136 miliónov návštev, z ktorých sme vyvodili tri kategórie článkov:

- Krátkodobé články
- Strednodobé články
- Dlhodobé články

Krátkodobé články boli charakteristické veľmi krátkym obdobím, do ktorého spadali všetky prístupy, pričom toto obdobie bolo často menej ako 24 hodín, po ktorých boli zaznamenané iba ojedinelé prístupy. Tieto články sa väčšinou týkali neočakávaných udalostí a náhlych správ. Strednodobé články vykazovali vzory striedajúcich sa nárastov a poklesov početnosti návštev, ktorá sa rozprestierala v dĺžke niekoľko dní až týždňov, po čom sa návštevnosť úplne vytratila ako v prípade krátkodobých článkov. Charakteristickou črtou strednodobých článkov bolo to, že sa týkali rovnako prebiehajúcich udalostí ako napríklad voľby, šampionát a podobne. Dlhodobé články vykazovali približne rovnomernú početnosť prístupov počas dlhého obdobia niekoľkých mesiacov a ich charakteristickou črtou bolo, že sa netýkali žiadnej časovo obmedzenej udalosti ale ich obsahom boli napríklad recepty, zdravý životný štýl a podobne.

## 4 Metóda pre odporúčanie akcií na základe situačných pravidiel

Hlavným cieľom pri návrhu našej metódy je poskytovanie podpory pre rozhodovanie na základe situácií o autonómne vykonávaných akciách v rámci služieb a aplikácií pre koncových používateľov. Našu metódu sme preto navrhli tak, aby nebola viazaná na špecifickú doménu a mohla byť použiteľná v rôznych oblastiach. Napriek tomu, že čas a poloha sú najčastejšie využívané súčasti kontextu, stále potrebujeme spôsob, akým sme schopní zachytiť špecifické typy kontextu pre špecifické domény, a preto sme sa rozhodli pre symbolickú reprezentáciu situácií ako aj pravidiel, ktoré slúžia na výpočet odporúčaní.

### 4.1 Model situácií

Model situácií pozostáva z množiny situácií, ktoré opisujú reálnu situáciu používateľa a prostredie v ktorom sa nachádza. Každá situácia je priradená do jednej z tried situácií, pričom tak triedy aj samotné situácie v nich sú definované klientskou službou alebo aplikáciou podľa ich špecifických potrieb. Takáto symbolická reprezentácia umožňuje reprezentovať aj komplexný kontext používateľa relatívne jednoduchou množinou reťazcov. Napríklad pre situáciu jasného počasia by sme definovali symbol

Počasie : Jasno

kde *Počasié* reprezentuje triedu a *Jasno* situáciu v rámci nej. Inštancie takýchto situácií potom tvoria model situácií používateľa v čase, pričom týmito inštanciami hovoríme pozorovania. Každé pozorovanie okrem symbolu identifikujúceho situáciu obsahuje ďalšie tri informácie:

- Čas
- Istota
- Miera zániku istoty

Čas určuje, pre aký časový okamih sa pozorovanie vzťahuje, istota vyjadruje ako istá si je klientska služba či aplikácia týmto pozorovaním a miera zániku istoty určuje ako rýchlo v čase stráca pozorovanie svoju základnú istotu, pričom vzorec pre úpravu istoty v čase je:

$$CF_t = \frac{CF_b}{(1 + r)^t}$$

kde  $CF_t$  je výsledná istota v čase  $t$ ,  $CF_b$  je pôvodne priradená istota,  $r$  je miera zániku istoty a  $t$  je čas v hodinách, ktorý prešiel od času pozorovania.

## 4.2 Model akcií

Model akcií pozostáva z množiny pravidiel, ktoré sú automaticky generované na základe pozorovaní indikátorov akcií definovaných klientskou aplikáciou. Indikátor akcie v špecifickej doméne je čokoľvek, čo indikuje vhodnosť situácie pre vykonanie určitej akcie. Napríklad v doméne internetových noviniek je veľmi silným indikátorom pre odporúčenie noviniek ak si používateľ práve číta nejaké novinky. Podobne situácia, v ktorej sa používateľ hrá hru môže byť tiež považovaná za vhodný indikátor, pretože pravdepodobne môže byť v takejto situácii vyrušený zaujímavou novinkou.

Každé pravidlo pozostáva z dvoch častí. Prvá časť je množina predpokladov, ktoré definujú, v akej situácii sa pravidlo aplikuje a druhá časť je akcia, ktorú pravidlo naznačuje. Každý z množiny predpokladov je vyjadrený symbolom zodpovedajúcim jednému zo symbolov modelu situácie a má priradenú istotu korešpondujúcu s istotou pozorovania danej situácie v čase, keď bolo pravidlo definované. Záver každého pravidla je akcia, opäť vyjadrená symbolom, ku ktorej je priradený faktor istoty a základná miera zániku istoty pravidla. Istota vytvorených pravidiel taktiež v čase zaniká no výsledná miera istoty je vypočítavaná zo základnej miery zániku istoty na základe toho, ako často sa od definovania pravidla pozorovali situácie z jeho predpokladov.

### 4.3 Tvorba pravidiel

Tvorba pravidiel prebieha na základe pozorovaní indikátorov akcií, pričom pri každom pozorovaní sa definuje jedno a viac pravidiel. Definícia pravidiel na základe pozorovania prebieha v troch krokoch. V prvom kroku sa definujú pravidlá so svojimi predpokladmi tak, že každé pravidlo má práve jeden predpoklad z každej triedy situácií, ktoré sú obsiahnuté v aktuálnom modeli situácie. Týmto spôsobom sú vytvorené pravidlá pre všetky kombinácie situácií aktuálneho modelu.

V druhom kroku je základná istota pozorovania distribuovaná medzi vytvorené pravidlá na základe pomerov istôt predpokladov nasledovne:

$$CF_r = CF_b \left( \frac{\sum CF_{a_r}}{\sum CF_a} \right)$$

kde  $CF_r$  je výsledná istota pravidla,  $CF_b$  je základná istota,  $\sum CF_{a_r}$  je súčet istôt predpokladov daného pravidla a  $\sum CF_a$  je súčet istôt predpokladov všetkých nových pravidiel.

Tretím a posledným krokom je zahrnutie nových pravidiel do modelu akcií a to tak, že pokiaľ neexistuje pravidlo s rovnakou signatúrou (rovnaká množina predpokladov aj záver), tak je pravidlo priamo pridané do modelu a pokiaľ takéto pravidlo už existuje tak je istota existujúceho pravidla pripočítaná k istote nového pravidla podľa nasledujúceho vzorca:

$$CF_a = \max \left( CF_{ot}, CF_r + (1 - CF_r) \cdot \left( CF_{ot} \cdot \left( \frac{CF_{ob} - CF_{ot}}{CF_{ob}} \right)^r \right) \right)$$

kde  $CF_a$  je upravená istota,  $CF_r$  je pôvodná istota nového pravidla,  $CF_{ob}$  je základná istota pôvodného pravidla a  $CF_{ot}$  je istota pôvodného pravidla v čase definície nového. Cieľom tohto modelu je zaistiť, aby sa silno neagregovali pravidlá vytvorené v rovnakom alebo blízkyh pozorovaniach, ale aby sa veľká časť zostávajúcej istoty pravidla agregovala s istotou nového pravidla podľa toho, ako staré pravidlo vydržalo test času.

### 4.4 Výpočet odporúčaní

Výpočet odporúčaní vychádza z pravidiel, ktoré sú v čase definované a skladá sa z troch krokov:

1. Výpočet dôležitosti tried situácií
2. Úprava istôt pravidiel

### 3. Výpočet výsledných odporúčaní

#### 4.4.1 Výpočet dôležitosti tried situácií

Dôležitosť tried situácií je odvodená od toho, ako veľmi sa distribúcia situácií v pravidlách podobá rovnomernému rozdeleniu na základe nasledujúceho vzorca:

$$I_{SC} = \frac{\left( \frac{d_{s_1} + d_{s_2} + \dots + d_{s_m}}{m} \right)}{\left( \frac{(1 - r_{sc}) + (m - 1) \cdot r_{sc}}{m} \right)}$$

kde  $I_{SC}$  je vypočítaná istota triedy,  $d_{s_1} + d_{s_2} + \dots + d_{s_m}$  je súčet vzdialeností v reálnom rozdelení jednotlivých situácií aktuálnej triedy od pomyselného rovnomerného rozdelenia danej triedy,  $m$  je počet situácií a teda  $\left( \frac{d_{s_1} + d_{s_2} + \dots + d_{s_m}}{m} \right)$  je priemerná vzdialenosť situácií od pomyselného rovnomerného rozdelenia a  $\left( \frac{(1 - r_{sc}) + (m - 1) \cdot r_{sc}}{m} \right)$  je maximálna možná priemerná vzdialenosť.

#### 4.4.2 Úprava istôt pravidiel

Úprava istôt pravidiel sa skladá z troch krokov. Prvým krokom je zníženie základnej istoty podľa princípu straty istoty v čase uvedeného v sekcii 4.2. Druhým krokom je výpočet, ako nezhoda medzi istotami situácií jednotlivých predpokladov a istotami situácií aktuálneho modelu situácií pre jednotlivé predpoklady ovplyvní výslednú istotu pravidla, a to na základe vzorca:

$$L_a = L_{a_B} + (1 - L_{a_B}) \cdot (1 - I_{SC})^{p - (p-1) \cdot L_{a_B}^{\frac{1}{q}}}$$

$$L_{a_B} = \min \left( 1, \frac{CF_s}{CF_a} \right)$$

kde  $L_a$  je výsledný faktor vplyvu pre predpoklad  $a$ ,  $L_{a_B}$  je základný faktor vplyvu vypočítaný zo zhody medzi istotou predpokladu a istotou zodpovedajúcej situácie v aktuálnom modeli situácií,  $I_{SC}$  je vypočítaná dôležitosť triedy situácií do ktorej patrí situácia  $a$  a parametre  $p$  a  $q$  ( $p, q \geq 1$ ) určujú tvar a rýchlosť zmeny tvaru krivky mapujúcej základný faktor vplyvu na výsledný faktor vplyvu.

Posledným tretím krokom je úprava istoty pravidla podľa vypočítaných faktorov vplyvu:

$$CF_f = (F_{a_1} \cdot F_{a_2} \cdot \dots \cdot F_{a_n}) \cdot CF_r$$

kde  $C$  je výsledná istota pravidla,  $F$  sú faktory vplyvu jednotlivých predpokladov pravidla a  $C$  je pôvodná istota pravidla (už upravená podľa princípu znižovania istoty pravidiel v čase).

#### 4.4.3 Výpočet výsledných odporúčaní

Pri výpočte výsledných odporúčaní sa zoskupia pravidlá do skupín podľa spoločných záverov a ich upravené istoty sa spočítajú podľa vzorca navrhnutého Davidom McAllisterom pre sčítavanie faktorov istoty:

$$CF = CF_a + CF_b(1 - CF_a)$$

kde  $CF_a$  a  $CF_b$  sú dva kladné faktory istoty.

### 4.5 Charakteristiky metódy a diskusia

Našu metódu sme navrhli tak, aby bola jednoducho aplikovateľná v akejkoľvek doméne. Jeden z dôležitých konceptov je princíp znižovania istoty pravidiel v čase, ktorý zabezpečuje možnosť zmeny modelu v prípade, že sa zmenia preferencie a správanie používateľa.

Jeden zo zásadných problémov pri práci s množstvom situácií je výber podstatných atribútov čo v našom prípade predstavujú triedy situácií. Vo všeobecnosti, keď automatizované metódy pracujú s príliš veľa nepodstatnými atribútmi, tak majú problém podávať kvalitné výsledky, pretože jednoducho pracujú s príliš veľkým množstvom nepodstatných dát. Naša metóda v tomto prípade dokáže pracovať efektívne, keďže pravidlá v rôznych situáciách sa agregujú s ohľadom na istoty jednotlivých tried vypočítaných z reálnych distribúcií situácií v pravidlách.

Okrem kladných faktorov istoty je taktiež možné pracovať aj s negatívnymi hodnotami, ktoré by znižovali výsledné hodnoty odporúčaní, no naše odporúčanie je radšej vytvoriť protikladnú aktivitu, nakoľko je tento prístup transparentnejší a dáva väčší priestor pre rozhodovanie o vykonaní akcií.

## 5 Metóda pre odporúčanie akcií v doméne internetových noviniek

### 5.1 Situácie

Sú špecifické situácie, ktoré do určitej miery priamo indikujú či je alebo nie je vhodné používateľovi podsunúť nejaké novinky. Ako sme už uviedli tieto slúžia práve ako indikátory pre akcie odporúčania. Čo nás viac zaujíma sú práve situácie, o ktorých vhodnosti či nevhodnosti pre proaktívne odporúčanie nevieme robiť explicitné uzávery, pretože tieto sa líšia od používateľa k používateľovi. Takýmito sú napríklad situácie z nasledovných tried:

- Poloha a pohyb
- Čas v dni
- Deň v týždni
- Týždeň v mesiaci
- Mesiac v roku
- Počasie
- Udalosti v kalendári
- Budík
- Sedenia
- Stav displeja zariadenia

Situácie v uvedených a ďalších triedach existujú v globálnej a viazanej forme. V globálnej forme sú situácie všeobecne rozpoznateľné a pozorovateľné všetkými zariadeniami, ako napríklad *Poloha:VPohybe*. Na druhej strane viazané situácie sú tie, ktoré sú identifikované pre konkrétneho používateľa, ako napríklad *Poloha:ZDomuDoPrace*. Aj keď takúto situáciu bude mať zrejme viacero používateľov, pre každého z nich je jej sémantický význam iný a preto každá vystupuje zvlášť pre používateľa ku ktorému sa viaže. Na druhej strane situácia *Poloha:VPohybe* opisuje stav pohybu medzi neznámymi lokalitami, čo je sémanticky rovnaké pre všetkých používateľov špecifickej klientskej služby či aplikácie.

### 5.2 Akcie a indikátory

Popri situáciách, pre ktoré nevieme priamo určiť ich vhodnosť či nevhodnosť pre proaktívne odporúčanie noviniek sú aj situácie, pre ktoré to do určitej miery vieme a tieto potom nazývame indikátory akcií. Pre doménu internetových noviniek sme definovali štyri rôzne indikátory pre odporúčanie noviniek a tri, ktoré indikujú nevhodnosť situácie. V tabuľke 1 je uvedený ich prehľad pričom sú zoradené od najvyššej istoty po najnižšiu.



**Tabuľka 1.** Indikátory vhodnosti/nevhodnosti situácie pre proaktívne odporúčanie noviniek.

Indikátory vhodnej situácie	Indikátory nevhodnej situácie
Používateľ si číta novinky	Používateľ zruší našu notifikáciu
Používateľ je v aplikácii pre zábavu (napríklad hru)	Používateľ ignoruje našu notifikáciu
Používateľ je v aplikácii pre prácu (napríklad textový procesor)	Používateľ ignoruje prichádzajúci hovor
Používateľ telefonuje	

Čítanie noviniek je pre väčšinu ľudí postrannou aktivitou, ktorá nie je hlavnou súčasťou ich práce či priamo ich koníček. Hľadanie vhodných situácií pre proaktívne odporúčanie noviniek je preto vo svojej podstate hľadanie vhodných situácií, kedy môže byť používateľ vyrušený s niečím, čo nevyhnutne nesúvisí s tým, čo práve robí. Domény ako napríklad sociálne siete, informácie o filmoch, citáty, vtipy, videá a podobný obsah vykazujú podobné charakteristiky (sú postrannými aktivitami), a teda princípy pre internetové noviny platia rovnako pre tieto a rovnako s nimi môže byť aj narábané.

## 6 Vyhodnotenie metódy pre odporúčanie akcií

Pre účely vyhodnotenia našej metódy sme implementovali simulačný framework a mobilnú aplikáciu pre odporúčanie noviniek, pričom sa celý náš experiment skladal z troch fáz.

V prvej fáze sme definovali rôzne simulácie prostredia a simulácie používateľov s rôznymi preferenciami a správaním, aby sme overili správnosť matematického modelu na pozadí a schopnosť našej metódy spoľahlivo identifikovať vhodné situácie pre vykonávanie akcií v jednoduchých ale aj zložitejších scenároch.

V druhej fáze sme distribuovali zjednodušenú verziu našej aplikácie pre odporúčanie noviniek šiestim používateľom, aby sme overili predpoklady experimentov z prvej fázy, a teda to, že spôsob akým sme simulovali prostredie a virtuálnych používateľov zodpovedá tomu ako to aj v skutočnosti je.

V tretej fáze sme zobrali poznatky zo živého experimentu a vykonali sme dodatočné simulácie s dôležitými charakteristikami z reálnej prevádzky, ktorými sme overovali schopnosť našej metódy efektívne pracovať aj v týchto podmienkach.

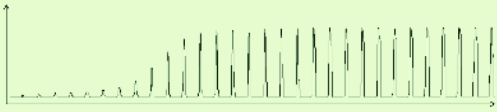
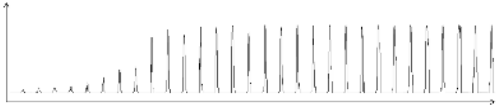

## 6.1 Základné simulácie

Základné simulácie sme rozdelili do dvoch skupín. Prvá skupina predstavuje jednoduché scenáre a druhá komplexné scenáre. Všetky simulácie, ktoré sme vykonávali brali pre výpočet odporúčaní v čase iba pravidlá, ktoré boli vytvorené minimálne dvanásť hodín spätne, aby sme zamedzili efektu výpočtu odporúčaní pre situácie na základe pravidiel, ktoré boli vytvorené práve na základe danej situácie.

### 6.1.1 Jednoduché scenáre

Hlavným cieľom simulácií jednoduchých scenárov bolo overiť matematický model našej metódy a celkovo schopnosť identifikácie vhodných okamihov. Výsledky a charakteristiky troch simulácií jednoduchých scenárov sa nachádzajú v tabuľke 2.

**Tabuľka 2.** Výsledky simulácií jednoduchých scenárov.

Výsledok simulácie	Čas	Opis
	30 dní	<ul style="list-style-type: none"> <li>- Jedna trieda situácií (čas v dni)</li> <li>- Používateľ preferujúci čítať novinky každé ráno</li> </ul>
	30 dní	<ul style="list-style-type: none"> <li>- Dve triedy situácií (čas v dni a počasie)</li> <li>- Trieda počasie obsahovala dve rôzne situácie</li> <li>- Používateľ preferujúci čítať novinky každé ráno</li> </ul>
	30 dní	<ul style="list-style-type: none"> <li>- Dve triedy situácií (čas v dni a deň v týždni)</li> <li>- Trieda deň v týždni obsahovala sedem rôznych situácií</li> <li>- Používateľ preferujúci čítať novinky každé ráno</li> </ul>



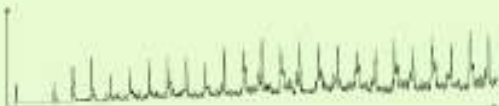

Výsledky simulácií jednoduchých scenárov ukazujú na korektnosť matematického modelu našej metódy v týchto podmienkach. Je vidieť, že s pridaním nepodstatnej triedy situácie pri druhej simulácii sa naša metóda iba nepatrne dlhšie prispôbuje simulovanému používateľovi a v prípade rozšírenia nepodstatnej triedy situácií pri tretej simulácii je scenár podobný. Vo všetkých troch prípadoch sú však jasne rozpoznateľné lokálne maximá, ktoré reprezentujú

našou metódou identifikované najvhodnejšie situácie pre odporúčanie, a ktoré korešpondujú s rákami počas trvania simulácií.

### 6.1.2 Komplexné scenáre

Hlavným cieľom komplexných scenárov bolo overiť schopnosť našej metódy správne identifikovať situácie vhodné pre odporúčanie akcií v komplexných simulovaných prostrediach pri rôznych typoch virtuálnych používateľov. Výsledky a charakteristiky štyroch simulácií komplexných scenárov sa nachádzajú v tabuľke 3.

**Tabuľka 3.** Výsledky simulácií komplexných scenárov.

Výsledok simulácie	Čas	Opis
	30 dní	- Deväť tried situácií - Používateľ preferujúci čítať novinky každé ráno
	90 dní	- Deväť tried situácií - Používateľ preferujúci čítať novinky pondelkové rána
	90 dní	- Deväť tried situácií - Používateľ preferujúci čítať novinky pondelkové rána a piatkové večery
	90 dní	- Deväť tried situácií - Používateľ preferujúci čítať novinky pondelkové rána so zmenou na piatkové večery

Jednotlivé simulácie na seba nadväzujú a z výsledkov je vidieť, že si naša metóda dokáže poradiť aj v prípadoch, kedy je až osem nepodstatných tried situácií a iba jediná podstatná (prvá simulácia), v scenári kde sú podstatné situácie iba ak sú nastanú naraz (druhá simulácia), v scenári kde je viac podstatných dvojíc situácií (tretia simulácia) ale aj v prípade, že príde k zmene preferencií (štvrtá simulácia).

## 6.2 Živý experiment

Počas živého experimentu podávali šiesti používatelia, o ktorých sme zbierali kontextové informácie, spätnú väzbu indikujúcu, či je alebo nie je vhodná situácia na čítanie noviniek alebo podobnú aktivitu. Spätnú väzbu bolo možné dať v štyroch stupňoch:

- Určite áno
- Skôr áno
- Skôr nie
- Určite nie

Počas dvoch týždňov sme od týchto šiestich používateľov nazbierali vyše 700 000 pozorovaní situácií a vyše 3 000 rôznych spätných väzieb, ktoré nám odhalili dve podstatné charakteristiky v reálnej prevádzke.


Prvou takouto charakteristikou je tzv. uzamknutie situácie, ku ktorému dochádza pri triedach situácií u ktorých sa situácia dlhodobejšie nemení. Takouto situáciou je napríklad teplota pri počasí, ktorá má obdobia stability. V takomto prípade potom dochádza k tomu, že pravidlá sú vytvárané s jedinou (uzamknutou) situáciou čo v našom prípade, kedy nepoznáme význam symbolov, spôsobí pri odomknutí situácie náhly pokles v istote odporúčaní. Tieto sa však po čase vrátia na svoju pôvodnú hodnotu, pretože metóda na základe nových spätných väzieb správne identifikuje takéto nepodstatné triedy ako nepodstatné.


Druhou identifikovanou charakteristikou sú náhodné indikácie, čo sú indikácie, ktoré sa vyskytnú ojedinele alebo veľmi zriedka a nepredstavujú súčasť žiadnej väzby situácia – akcia pri množine tried situácií, ktoré sú k dispozícii (sú pozorované). Je možné, že táto spätná väzba je závislá na nejakej súčasti kontextu používateľa, s ktorým naša metóda nepracuje, pretože ho klientska služba či aplikácia nesleduje, no v oboch prípadoch je dôležité, aby naša metóda dokázala popri tejto náhodnosti dostatočne jednoznačne identifikovať aspoň tie najvhodnejšie okamihy, ktoré závisia na sledovanej časti kontextu používateľa.

## 6.3 Doplnujúce simulácie

Cieľom doplnujúcich simulácií bolo overiť, ako naša metóda zareaguje, pokiaľ doplníme do simulácií charakteristiky reálneho prostredia pozorované počas živého experimentu. Výsledky oboch simulácií a ich charakteristiky sú uvedené v tabuľke 4.

**Tabuľka 4.** Výsledky doplnujúcich simulácií.

Výsledok simulácie	Čas	Opis
	42 dní	- Prvé dva týždne jediná situácia teploty ovzdušia - V týždňoch tri a štyri pridané ďalšie dve susedné

		situácie - V týždňoch päť a šesť pridaná nová situácia a odstránená situácia z prvých dvoch týždňov
	90 dní	- 7 až 14 náhodných indikácií (spätných väzieb) v každom týždni

Výsledky dopĺňujúcich simulácií ukazujú, že naša metóda je schopná korektného fungovania aj v prípade scenárov s náročnejšími charakteristikami, akými sú napríklad uzamknutie situácie či náhodné indikácie. V prípade uzamknutej situácie (prvá simulácia) je vidieť po dvoch a štyroch týždňoch pokles istoty odporúčaní ako sme aj predpokladali, no vhodné situácie (reprezentované lokálnymi maximami) sú stále jednoducho a spoľahlivo identifikovateľné. V prípade náhodných indikácií napriek tomu, že spätná väzba, ktoré je pre nás zaujímavá tvorí skoro až iba jednu tretinu celkovej spätnej väzby, tak sme opäť stále schopní správnej identifikácie vhodných situácií.

## 7 Záver a budúca práca

Žijeme vo svete všadeprítomných počítačov, kde sa automatizované technológie stávajú súčasťou nášho každodenného života. Väčšina týchto technológií však dnes pracuje na princípe jednotnosti prístupu, kde nerozlišujú rozdiely medzi jednotlivými používateľmi. Podobný problém je prítomný napríklad aj v doméne proaktívneho odporúčania noviniek alebo podobného zaujímavého obsahu, kde metódy zvyčajne zvažujú iba obsah a nie či je napríklad z pohľadu používateľa ten správny čas.

Navrhli sme a v tejto práci opísali našu metódu pre podporu autonómnosti pervasívnych služieb a aplikácií ich obohatením o odporúčanie vhodnosti vykonania špecifickej akcie na základe pozorovaní a získanej spätnej väzby.

Vyhodnotenie, ktoré sme vykonali poukazuje na to, že naša metóda je schopná pracovať v širokej množine prípadov, od takých, ktoré potrebujú iba malú časť kontextu používateľa až po také, kde používatelia zmenia svoje preferencie.

Naša budúca práca zahŕňa najmä vylepšovanie schopnosti našej metódy poradiť si so skrytými premennými a prepojeniami medzi nimi. To zahŕňa redukciu šumu, identifikáciu stabilných tried situácií a charakteristík tried situácií vo všeobecnosti pre lepšiu aproximáciu ich dôležitosti. Jedným

z otvorených problémov je taktiež identifikácia podmodelov v rámci modelu akcií. Ďalšie vylepšenia zahŕňajú napríklad podporu pre kolaboratívne modely, kde môžu byť pravidlá zdieľané medzi podobnými používateľmi podobne ako je aj obsah odporúčaný pri kolaboratívnom odporúčaní. Takáto podpora kolaboratívneho modelu by potom mohla v prípade už existujúcej širšej bázy používateľov významne znížiť problémy studeného štartu pre nových používateľov.

Metódu pre odporúčanie akcií sme navrhli tak, aby bola jednoducho aplikovateľná v akejkoľvek doméne a jedinou zodpovednosťou, ktorú majú služby a aplikácie pre koncových používateľov je rozhodnutie o tom, ktoré akcie budú vykonávať a aký kontext môže byť pre rozhodovanie o týchto akciách relevantný. Veríme, že naša metóda a pracovné rámce na nej postavené povzbudia vďaka svojej jednoduchosti viac služieb a aplikácií k tomu aby v sebe zahŕňali dôležitý kontext používateľa a pôsobili vďaka tomu inteligentnejšie.

## Referencie

1. Simpson, J., Weiner, E., eds.: The Oxford English Dictionary 2nd edn. Oxford University Press, Oxford (1989)
2. Böhmer, M., Bauer, G.: Exploiting the Icon Arrangement on Mobile Devices as Information Source for Context-awareness. In : MobileHCI, Lisboa, pp.195-198 (2010)
3. Krüger, A., Baus, J., Heckmann, D., Kruppa, M., Wasinger, R.: Adaptive Mobile Guides. In : The Adaptive Web: Methods and Strategies of Web Personalization. Springer-Verlag, New York (2007)
4. Korpipää, P., Häkkinä, J., Kela, J., Ronkainen, S., Käsälä, I.: Utilising Context Ontology in Mobile Device Application Personalization. In : MUM, Maryland, pp.133-140 (2004)
5. Ala-Siuru, P., Tapani, R.: Understanding and recognizing usage situations using context data available in mobile phones. In : ubiPCMM, California (2006)
6. Roberts, M., Ducheneaut, N., Beloge, B., Partridge, K., Price, B., Bellotti, V., Walendowski, A., Rasmussen, P.: Scalable Architecture for Context-Aware Activity-Detecting Mobile Recommendation Systems. In : WoWMoM, California, pp.1-6 (2008)
7. Billsus, D., Pazzani, M.: Adaptive News Access. In : The Adaptive Web: Methods and Strategies of Web Personalization. Springer-Verlag, New York (2007)
8. Cantador, I., Castells, P.: Semantic Contextualisation in a News Recommender System. In : Workshop on Context-Aware Recommender Systems, New York (2009)
9. Liu, J., Dolan, P., Pedersen, E.: Personalized News Recommendation Based on Click Behavior. In : IUI, Hong Kong, pp.31-40 (2010)





# Appendix A:

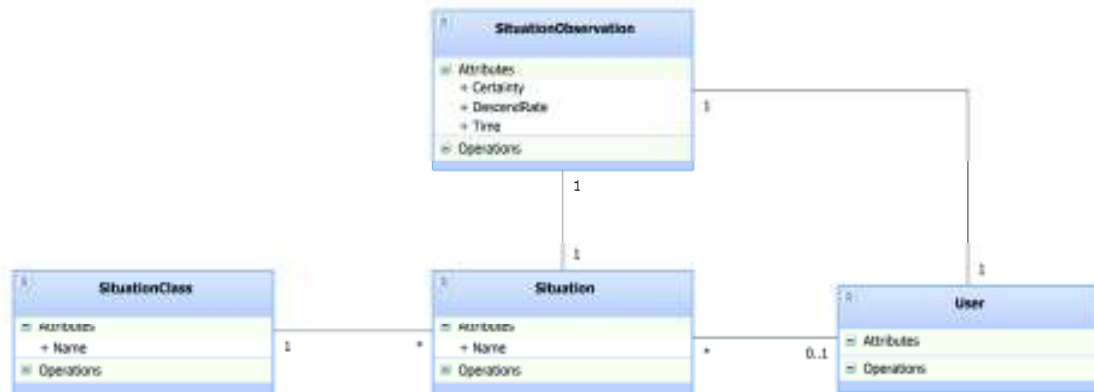
## Technical Documentation



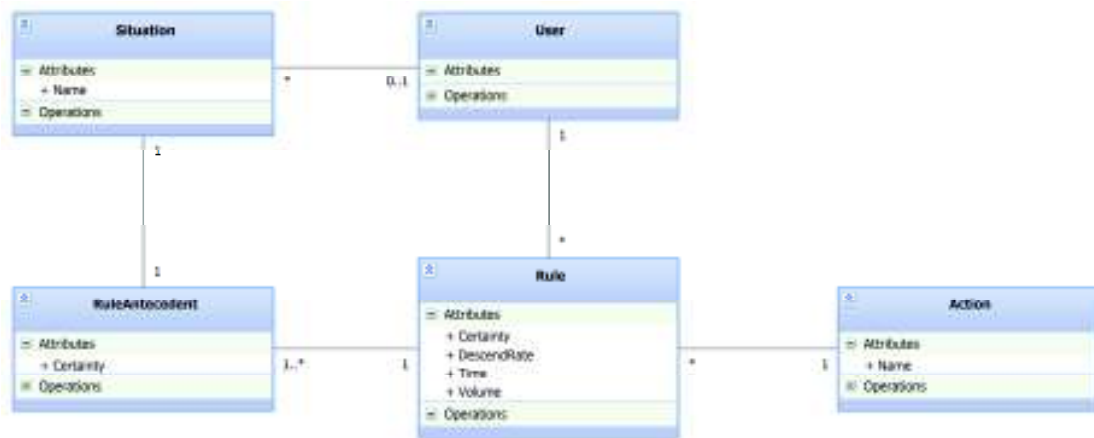
## User Models

Class diagrams in this section represent views on the important parts of the two interlinked models that are used for storage of situation classes, situations and their observations as well as storage for actions, indicator observations and rules that the recommendations are based on.

### Situation Model

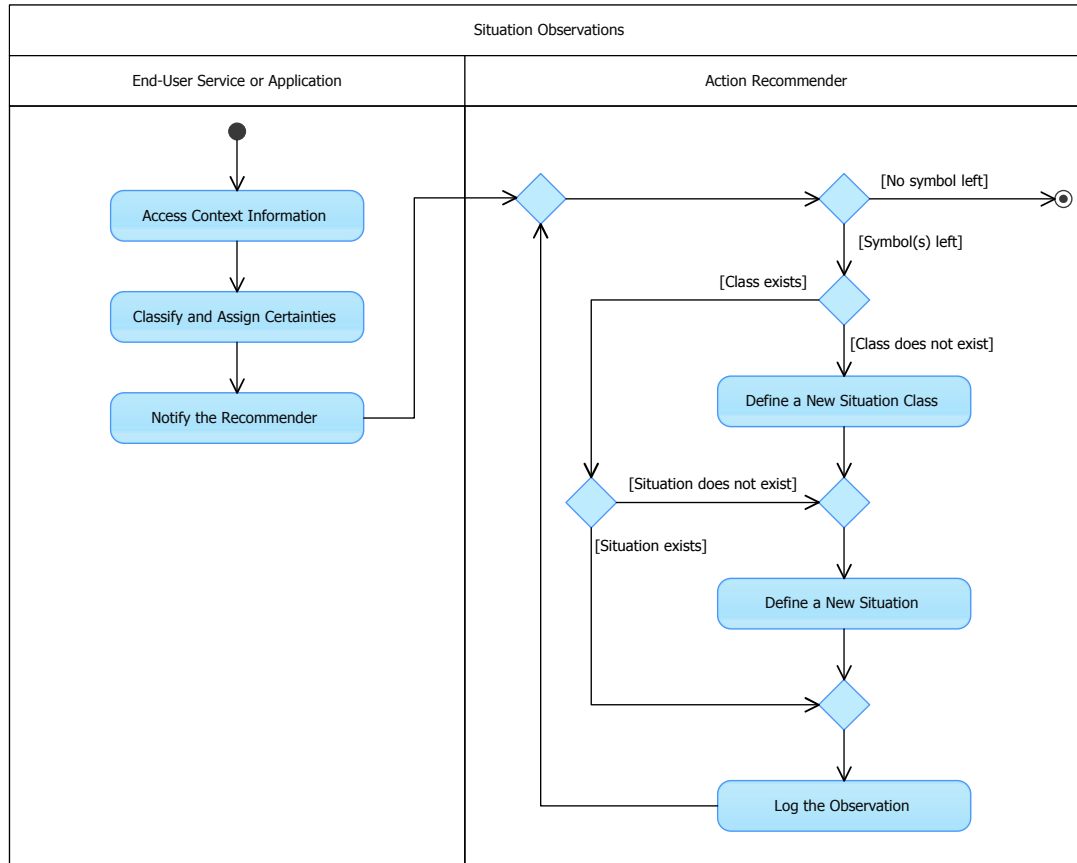


### Action Model

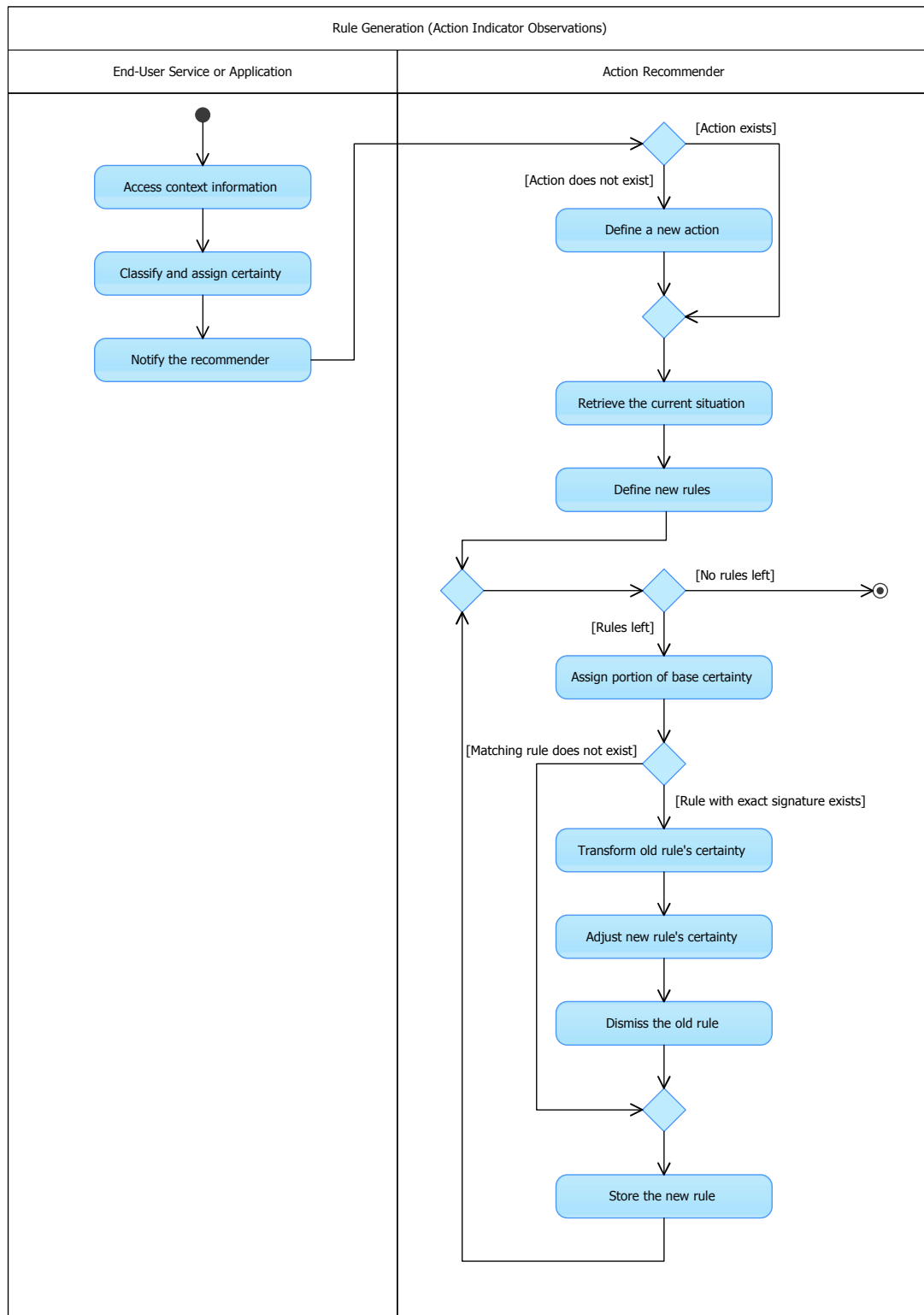


# Method Processes

## Observations and Rule Generation



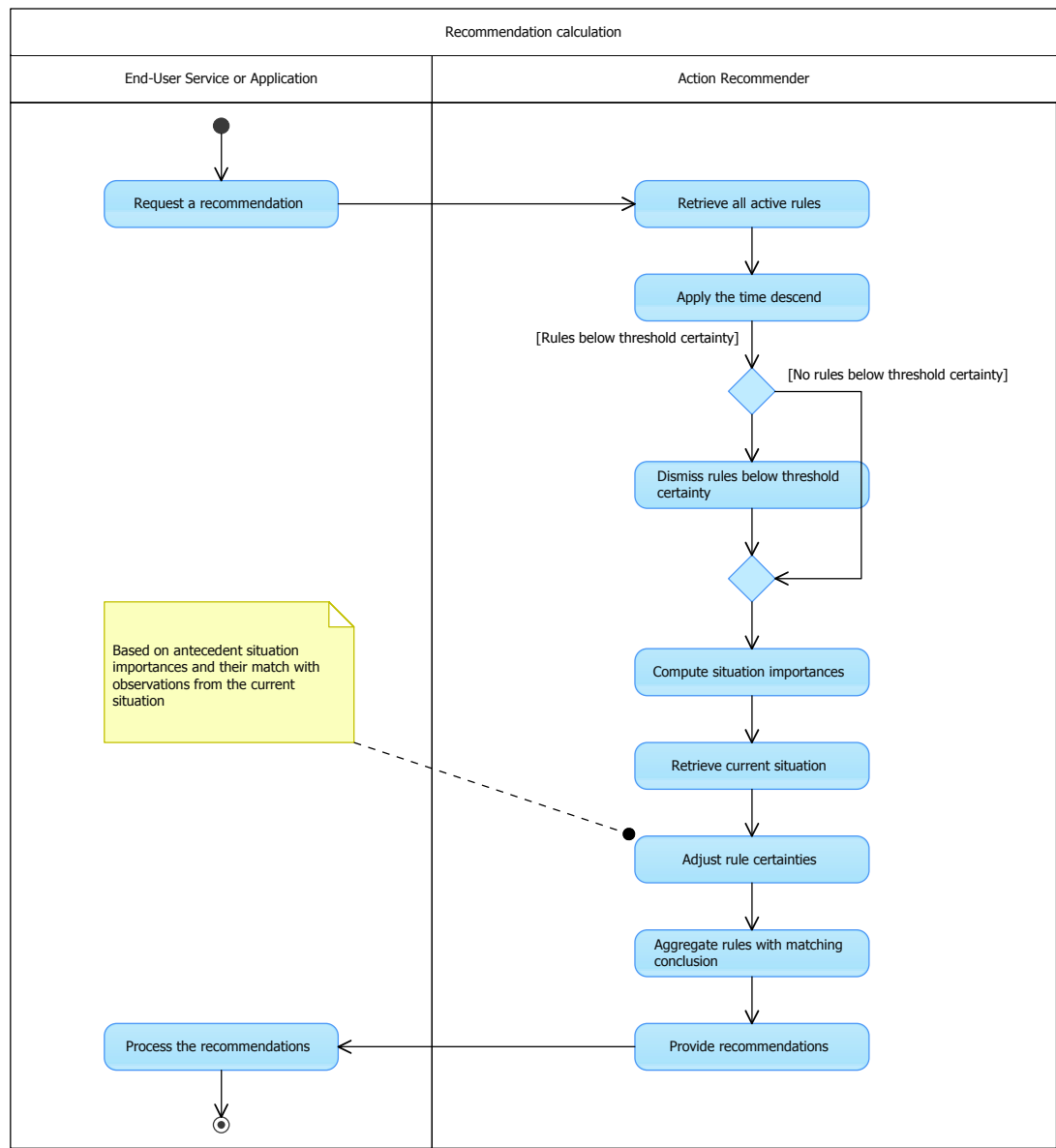
Situation observations are fed to our method at arbitrary times that depend mostly on the end-user service or application. They are fed as symbols carrying timestamp, certainty value and descend rate. Our method processes all symbols one by one defining situation classes and situations within them as necessary. Should be a user-specific (local) situation observed for the first time it has to be explicitly defined before its first observation can be fed in and must not have the same signature as any other global symbol so as to avoid treating it as one.



When observing action indicator new rules are defined based on all possible combinations among situation classes of situations from the current user's

situation (most recent observations of any situation class). All rules that have no match in the base are then simply added and rules that have an existing match are aggregated with it and then added to the persistent rule base.

Recommendation Calculation



The recommendations are based on all present and active rules where their certainty is first lowered according to the time sensitivity principle and then based on the computed antecedent importance and match between their antecedents and observations from the current situation model.

# Action Recommender Code Examples

## Rule Definition

```
[STORED PROCEDURE]
USP_CreateRules
(
    @id_ContextActionLog bigint
)

/* *****
/* Define rules for the provided action observation */
/* *****

/* Retrieve the context action log */
CREATE TABLE #ContextActionLog (id_action bigint, id_user bigint, certainty float, descendRate
float, timeRegistered datetime2(7))
INSERT INTO #ContextActionLog (id_action, id_user, certainty, descendRate, timeRegistered)
    SELECT id_contextAction, id_user, certainty, descendRate, timeRegistered
    FROM ContextActionLogs
    WHERE id = @id_ContextActionLog

/* Store the context action log data */
DECLARE @id_user bigint = (SELECT TOP(1) id_user FROM #ContextActionLog)
DECLARE @id_action float = (SELECT TOP(1) id_action FROM #ContextActionLog)
DECLARE @action_timeRegistered datetime2(7) = (SELECT TOP(1) timeRegistered FROM
#ContextActionLog)
DECLARE @action_certainty float = (SELECT TOP(1) certainty FROM #ContextActionLog)
DECLARE @action_descendRate float = (SELECT TOP(1) descendRate FROM #ContextActionLog)

/* Retrieve the current situation */
CREATE TABLE #CurrentSituation (id_situationClass bigint, id_situation bigint, id_log bigint,
certainty float, descendRate float, timeRegistered datetime2(7))
INSERT INTO #CurrentSituation (id_situationClass, id_situation, id_log, certainty, descendRate,
timeRegistered)
    SELECT id_situationClass, id_situation, id, certainty, descendRate, timeRegistered
    FROM dbo.UFN_GetCurrentSituation(@id_user, @action_timeRegistered)

/* Retrieve all defined situation classes and situations */
CREATE TABLE #Situations (id_situationClass bigint, id_situation bigint, unique clustered
(id_situationClass, id_situation))
INSERT INTO #Situations (id_situationClass, id_situation)
    SELECT id_situationClass, id_situation
    FROM #CurrentSituation
    ORDER BY id_situationClass, id_situation

/* Define the rules table and insert the initial empty rule */
CREATE TABLE #Rules (id bigint IDENTITY(1,1) NOT NULL, generation bigint, id_old bigint,
id_newSituationClass bigint, id_newSituation bigint, [signature] varchar(MAX), certainty float)

INSERT INTO #Rules (generation, [signature]) VALUES (0, '')

/* Fill the rules table */
DECLARE @id_situationClass bigint
DECLARE @generation bigint = 1
WHILE ((SELECT COUNT(*) FROM #Situations) > 0)
BEGIN

    SET @id_situationClass = (SELECT TOP(1) id_situationClass FROM #Situations)
    DELETE FROM #Situations WHERE id_situationClass = @id_situationClass

    INSERT INTO #Rules (generation, id_old, id_newSituationClass, id_newSituation,
[signature])
        SELECT @generation, ContainedRules.id, #CurrentSituation.id_situationClass,
```

```

#CurrentSituation.id_situation, ContainedRules.[signature] + CAST(#CurrentSituation.id_situation
AS varchar(MAX)) + ';'
        FROM #CurrentSituation
        LEFT JOIN #Rules AS ContainedRules ON 1 = 1
        WHERE #CurrentSituation.id_situationClass = @id_situationClass
              AND ContainedRules.generation = @generation - 1

        SET @generation = @generation + 1
END

/* Store the last generation number and set the iterator */
DECLARE @maxGeneration bigint = @generation - 1
SET @generation = @maxGeneration

/* Define the rule antecedents table */
CREATE TABLE #RuleAntecedents (id_rule bigint, id_situationClass bigint, id_situation bigint,
certainty float, timeRegistered datetime2(7))

/* Store rule antecedents */
WHILE ((SELECT COUNT(*) FROM #Rules WHERE #Rules.generation = @maxGeneration AND #Rules.id_old
IS NOT NULL) > 0)
BEGIN

        INSERT INTO #RuleAntecedents (id_rule, id_situationClass, id_situation, certainty,
timeRegistered)
        SELECT #Rules.id, #Rules.id_newSituationClass, #Rules.id_newSituation,
#CurrentSituation.certainty, #CurrentSituation.timeRegistered
        FROM #Rules
        LEFT JOIN #CurrentSituation ON #Rules.id_newSituation =
#CurrentSituation.id_situation
        WHERE #Rules.generation = @maxGeneration

        UPDATE #Rules
        SET id_newSituationClass = (SELECT id_newSituationClass FROM #Rules AS Nested
WHERE Nested.id = #Rules.id_old),
        id_newSituation = (SELECT id_newSituation FROM #Rules AS Nested WHERE
Nested.id = #Rules.id_old)
        WHERE #Rules.generation = @maxGeneration

        UPDATE #Rules
        SET id_old = (SELECT id_old FROM #Rules AS Nested WHERE Nested.id =
#Rules.id_old)
        WHERE #Rules.generation = @maxGeneration

END

/* Delete all intermediate rules */
DELETE FROM #Rules WHERE generation < @maxGeneration

/* Distribute rule certainties */
UPDATE #Rules
SET certainty =
(
        @action_certainty
        *
        (
                (SELECT SUM(#RuleAntecedents.certainty) FROM #RuleAntecedents WHERE
#RuleAntecedents.id_rule = #Rules.id)
                /
                (SELECT SUM(#RuleAntecedents.certainty) FROM #RuleAntecedents)
        )
)

/* Copy all rules to the persistent base */
WHILE ((SELECT COUNT(*) FROM #Rules) > 0)
BEGIN

        /* Get information about the new rule*/
        DECLARE @tempRule_id bigint = (SELECT TOP(1) id FROM #Rules)
        DECLARE @tempRule_signature varchar(MAX) = (SELECT [signature] FROM #Rules WHERE id =

```



```

@tempRule_id)

    /* Get information about an existing rule with the same signature */
    DECLARE @existingRule_id bigint = (SELECT TOP(1) id FROM Rules WHERE (id_user =
@id_user) AND ([signature] = @tempRule_signature) AND (timeCeased IS NULL) AND (timeRegistered
<= @action_timeRegistered))
    DECLARE @existingRule_baseCertainty float = 0.0
    DECLARE @existingRule_ceasedCertainty float = 0.0
    DECLARE @existingRule_certainty float = 0.0
    DECLARE @existingRule_volume float = 0.0

    /* If there is an existing active rule with the same signature then retrieve its
certainty */
    IF (@existingRule_id IS NOT NULL)
    BEGIN
        DECLARE @baseAntecedentCeaseRate float = (SELECT [value] FROM [Parameters] WHERE
[name] = 'BaseAntecedentCeaseRate')
        SET @existingRule_baseCertainty = (SELECT certainty FROM Rules WHERE id =
@existingRule_id)
        SET @existingRule_ceasedCertainty = dbo.UFN_EaseRuleCertainty(@existingRule_id,
@action_timeRegistered, @baseAntecedentCeaseRate, 1)

        DECLARE @existingRule_ceaseValue float = (@existingRule_baseCertainty -
@existingRule_ceasedCertainty)
        SET @existingRule_certainty = @existingRule_ceasedCertainty *
(@existingRule_ceaseValue / @existingRule_baseCertainty)

        SET @existingRule_volume = (SELECT volume FROM Rules WHERE id =
@existingRule_id)

        /* Also incorporate the antecedent certainties from the previous matching rule
*/
        UPDATE #RuleAntecedents
        SET certainty =
        (
            #RuleAntecedents.certainty
            +
            (
                (
                    SELECT certainty
                    FROM RuleAntecedents
                    WHERE RuleAntecedents.id_rule =
@existingRule_id
AND
RuleAntecedents.id_situation = #RuleAntecedents.id_situation
                )
                *
                @existingRule_volume
            )
        )
        /
        (
            @existingRule_volume
            +
            1
        )
        WHERE #RuleAntecedents.id_rule = @tempRule_id

    END

    /* Cease the old rule */
    UPDATE Rules SET timeCeased = @action_timeRegistered WHERE id = @existingRule_id

    /* Define the new rule */
    INSERT INTO Rules (id_user, id_action, certainty, descendRate, [signature],
timeRegistered, volume, id_parent)
    SELECT @id_user, @id_action, dbo.UFN_Max(@existingRule_ceasedCertainty,
dbo.UFN_CFSum(#Rules.certainty, dbo.UFN_CFMul(@existingRule_certainty, @existingRule_volume))),
@action_descendRate, #Rules.[signature], @action_timeRegistered, @existingRule_volume + 1,
@existingRule_id
    FROM #Rules

```

```

WHERE #Rules.id = @tempRule_id

/* Store the persistent rule id */
DECLARE @newRule_id bigint = @@IDENTITY

/* Store antecedents of the new rule */
INSERT INTO RuleAntecedents (id_rule, timeRegistered, id_situationClass, id_situation,
certainty)
SELECT @newRule_id, @action_timeRegistered, #RuleAntecedents.id_situationClass,
#RuleAntecedents.id_situation, #RuleAntecedents.certainty
FROM #RuleAntecedents
WHERE #RuleAntecedents.id_rule = @tempRule_id

/* Delete the processed rule */
DELETE FROM #RuleAntecedents WHERE id_rule = @tempRule_id
DELETE FROM #Rules WHERE id = @tempRule_id

END

DROP TABLE #RuleAntecedents
DROP TABLE #Rules
DROP TABLE #Situations
DROP TABLE #CurrentSituation
DROP TABLE #ContextActionLog

```

## Time Sensitivity for Rules

```

[FUNCTION]
UFN_EaseRuleCertainty
(
    @id_rule bigint,
    @time datetime2(7),
    @baseAntecedentCeaseRate float,
    @testForTimeCeased bit = 1
),
/* ***** */
/* Apply time sensitivity principle to the provided rule */
/* ***** */

/* Test if the time is not behind the rule's time of cease */
IF (@testForTimeCeased = 1)
BEGIN
    DECLARE @timeCeased datetime2(7) = (SELECT timeCeased FROM Rules WHERE id = @id_rule)
    IF ((@timeCeased IS NOT NULL) AND (@timeCeased <= @time))
        RETURN 0.0
END

/* Get the base information about the rule and its owner */
DECLARE @id_user bigint = (SELECT id_user FROM Rules WHERE id = @id_rule)
DECLARE @timeRegistered datetime2(7) = (SELECT timeRegistered FROM Rules WHERE id = @id_rule)
DECLARE @certainty float = (SELECT certainty FROM Rules WHERE id = @id_rule)
DECLARE @baseCeaseRate float = (SELECT descendRate FROM Rules WHERE id = @id_rule)
DECLARE @ceaseRate float = 0.0

/* Get the rule antecedents */
DECLARE @RuleAntecedents TABLE (id_situationClass bigint, id_situation bigint, certainty float,
occurrence float)
INSERT INTO @RuleAntecedents (id_situationClass, id_situation, certainty)
SELECT Situations.id_situationClass, RuleAntecedents.id_situation,
RuleAntecedents.certainty
FROM RuleAntecedents
LEFT JOIN Situations ON Situations.id = RuleAntecedents.id_situation
WHERE id_rule = @id_rule

/* Get the rule antecedent occurrence */

```

```

UPDATE @RuleAntecedents
    SET occurrence =
    (
        SELECT SUM(dbo.UFN_Min(1.0, SituationLogs.clusterCertainty /
InnerSituations.certainty))
        FROM @RuleAntecedents AS InnerSituations
        LEFT JOIN SituationLogs ON InnerSituations.id_situation =
SituationLogs.id_situation
        WHERE SituationLogs.id_user = @id_user
        AND SituationLogs.id_situation =
[@RuleAntecedents].id_situation
        AND SituationLogs.timeRegistered >= @timeRegistered
        AND SituationLogs.timeRegistered < @time
        AND SituationLogs.clusterCertainty IS NOT NULL
    )
UPDATE @RuleAntecedents SET occurrence = 0.0 WHERE occurrence IS NULL

/* Compute the cease rate parameter */
SET @ceaseRate =
(
    SELECT @baseCeaseRate * AVG(1.0 * (occurrence / @baseAntecedentCeaseRate))
    FROM @RuleAntecedents
    --LEFT JOIN @SituationImportance ON [@SituationImportance].id_situation =
[@SituationClusterSums].id_situation
)

/* Compute the final certainty */
DECLARE @result float = dbo.UFN_EaseCertainty(@certainty, @ceaseRate, @timeRegistered, @time,
24.0)

RETURN @result

```

## Compute Recommendations

```

[STORED PROCEDURE]
USP_GetActionProposals
(
    @id_user bigint,
    @time datetime2(7)
),
/* *****
/* Get action suggestions for the provided user and time */
/* *****

    IF (@time IS NULL)
        SET @time = GETUTCDATE()

/* Get the current situation */
CREATE TABLE #CurrentSituation (id_situationClass bigint, id_situation bigint, id bigint,
certainty float, descendRate float, timeRegistered datetime2(7))
INSERT INTO #CurrentSituation (id_situationClass, id_situation, id, certainty, descendRate,
timeRegistered) (SELECT * FROM dbo.UFN_GetCurrentSituation(@id_user, @time))
INSERT INTO #CurrentSituation (id_situationClass, id_situation, certainty) (SELECT
id_situationClass, id, 0.0 FROM Situations WHERE id NOT IN (SELECT id_situation FROM
#CurrentSituation))

DECLARE @previousTime datetime2(7) = DATEADD(hour, -12, @time)

/* Get all eligible rules with their certainties eased */
CREATE TABLE #Rules (id bigint, ruleName nvarchar(64), certainty float, descendRate float,
timeRegistered datetime2(7))
--INSERT INTO #Rules (id, ruleName, certainty, descendRate, timeRegistered) (SELECT Rules.id,
ContextActions.name, Rules.certainty, Rules.descendRate, Rules.timeRegistered FROM Rules LEFT
JOIN ContextActions ON Rules.id_action = ContextActions.id WHERE Rules.id_user = @id_user AND

```

```

Rules.timeRegistered < @time AND (Rules.timeCeased IS NULL OR Rules.timeCeased > @time))
INSERT INTO #Rules (id, ruleName, certainty, descendRate, timeRegistered)
    --EXEC dbo.USP_GetRules @id_user, @time, 1
    EXEC dbo.USP_GetRules @id_user, @previousTime, 1 -- Only use rules defined at least 4
hours ago

/* Cease rule certainties */
--UPDATE #Rules SET certainty = dbo.UFN_EaseRuleCertainty(id, @time, 10.0)

/* Get all their antecedents */
CREATE TABLE #RuleAntecedents (id_rule bigint, id_situation bigint, certainty float)
INSERT INTO #RuleAntecedents (id_rule, id_situation, certainty) (SELECT id_rule, id_situation,
certainty FROM RuleAntecedents WHERE id_rule IN (SELECT id FROM #Rules))

--UPDATE #RuleAntecedents SET certainty = certainty * (SELECT certainty FROM #Rules WHERE
#Rules.id = #RuleAntecedents.id_rule)
--SELECT id_situation, SUM(certainty) FROM #RuleAntecedents WHERE id_situation IN (67, 68) GROUP
BY id_situation

/* Get the situation importances */
CREATE TABLE #SituationImportance(id_situationClass bigint, id_situation bigint,
situationImportance float)
INSERT INTO #SituationImportance (id_situationClass, id_situation, situationImportance)
    EXEC dbo.USP_GetSituationImportance @id_user, @time

/* Distribute the importance on the whole 0-1 interval */
UPDATE #SituationImportance
    SET situationImportance = (situationImportance - (SELECT MIN(situationImportance) FROM
#SituationImportance)) / ((SELECT MAX(situationImportance) FROM #SituationImportance) - (SELECT
MIN(situationImportance) FROM #SituationImportance))

--SELECT * FROM #SituationImportance ORDER BY id_situationClass

/* Get the model parameters */
DECLARE @baseInfluenceExponent float = (SELECT [value] FROM [Parameters] WHERE [name] =
'BaseInfluenceExponent')
DECLARE @baseInfluenceExponentChangeExponent float = (SELECT [value] FROM [Parameters] WHERE
[name] = 'BaseInfluenceExponentChangeExponent')

/* Update certainty to influence */
UPDATE #RuleAntecedents
    SET certainty =
    (
        dbo.UFN_Min
        (
            1.0,
            (
                (SELECT TOP(1) certainty FROM #CurrentSituation WHERE
#CurrentSituation.id_situation = #RuleAntecedents.id_situation)
                /
                certainty
            )
        )
    )

/* Update rule antecedents based on their importance */
UPDATE #RuleAntecedents
    SET certainty =
    (
        certainty
        +
        (1.0 - certainty)
        *
        POWER
        (
            (1.0 - (SELECT TOP(1) situationImportance FROM #SituationImportance
WHERE #SituationImportance.id_situation = #RuleAntecedents.id_situation)),
            @baseInfluenceExponent
    )

```

```

        (
            (@baseInfluenceExponent - 1.0)
            *
            POWER
            (
                certainty,
                (1.0 / @baseInfluenceExponentChangeExponent)
            )
        )
    )
)

/* Correct NULL certainty where present */
UPDATE #RuleAntecedents SET certainty = 0.0 WHERE certainty IS NULL

/* Update rule certainties based on antecedent certainty ratios */
UPDATE #Rules SET certainty = certainty * (SELECT dbo.Mul(certainty) FROM #RuleAntecedents WHERE
#RuleAntecedents.id_rule = #Rules.id)

/* Delete all rules that have null certainty */
DELETE FROM #Rules WHERE [certainty] IS NULL

/* Prepare the suggestions table */
CREATE TABLE #Suggestions (actionName nvarchar(64), certainty float)
INSERT INTO #Suggestions (actionName, certainty) (SELECT DISTINCT(ruleName),
dbo.CFSum(certainty) FROM #Rules GROUP BY ruleName)

/* Return the results */
SELECT @time AS [time], * FROM #Suggestions

/* Clean-up */
DROP TABLE #Suggestions
DROP TABLE #Rules
DROP TABLE #CurrentSituation
DROP TABLE #SituationImportance
DROP TABLE #RuleAntecedents

```

## Compute Situation Importance

```

[STORED PROCEDURE]
USP_GetSituationImportance
(
    @id user bigint,
    @time datetime2(7)
),
/*****
/* Get importance of all situations for the provided user and time */
*****/

/* Get all eligible rules */
CREATE TABLE #Rules (id bigint, ruleName nvarchar(64), certainty float, descendRate float,
timeRegistered datetime2(7));
INSERT INTO #Rules (id, ruleName, certainty, descendRate, timeRegistered)
SELECT id, ruleName, certainty, descendRate, timeRegistered FROM
dbo.UFN_GetRules(@id_user, @time, 0)

/* Get all their antecedents */
CREATE TABLE #RuleAntecedents (id_rule bigint, id_situation bigint, id_situationClass bigint,
certainty float)
INSERT INTO #RuleAntecedents (id_rule, id_situation, id_situationClass, certainty)
SELECT id_rule, id_situation, id_situationClass, certainty
FROM RuleAntecedents
WHERE id_rule IN
(

```

```

        SELECT id
        FROM #Rules
    )

/* Load all situations relevant to the current user. */
CREATE TABLE #SituationImportance (id_situationClass bigint, id_situation bigint,
classRuleCertaintySum float, situationRuleCertaintySum float, classRuleCertaintyCount float,
situationRuleCertaintyCount float, classSize bigint, activeClassSize bigint, classRatio float,
activeClassRatio float, situationRatio float, situationDistance float, activeSituationDistance
float, countRatio float, countDistance float, averageClassDistance float, maximumClassDistance
float, maximumActiveClassDistance float, situationImportance float)
INSERT INTO #SituationImportance (id_situationClass, id_situation)
    SELECT id_situationClass, id
    FROM Situations
    WHERE (id_owner IS NULL) OR (id_owner = @id_user)

/* Compute the situation certainty sum in user's relevant rules */
UPDATE #SituationImportance
    SET situationRuleCertaintySum =
    (
        SELECT SUM(certainty)
        FROM #RuleAntecedents
        WHERE id_situation = #SituationImportance.id_situation
    ),
    situationRuleCertaintyCount =
    (
        SELECT COUNT(certainty)
        FROM #RuleAntecedents
        WHERE id_situation = #SituationImportance.id_situation
    )

/* Compute the class certainty sum in user's relevant rules */
UPDATE #SituationImportance
    SET classRuleCertaintySum =
    (
        SELECT SUM(situationRuleCertaintySum)
        FROM #SituationImportance AS Nested
        WHERE #SituationImportance.id_situationClass = Nested.id_situationClass
    ),
    classRuleCertaintyCount =
    (
        SELECT SUM(situationRuleCertaintyCount)
        FROM #SituationImportance AS Nested
        WHERE #SituationImportance.id_situationClass = Nested.id_situationClass
    )

UPDATE #SituationImportance
    SET countRatio = situationRuleCertaintyCount / classRuleCertaintyCount
    WHERE classRuleCertaintyCount != 0

/* Compute the number of situations (both general and used) in each class */
UPDATE #SituationImportance
    SET classSize =
    (
        SELECT COUNT(id_situation)
        FROM #SituationImportance AS Nested
        WHERE #SituationImportance.id_situationClass = Nested.id_situationClass
    ),
    activeClassSize =
    (
        SELECT COUNT(id_situation)
        FROM #SituationImportance AS Nested
        WHERE (#SituationImportance.id_situationClass =
Nested.id_situationClass) AND (Nested.situationRuleCertaintySum IS NOT NULL)
    )
UPDATE #SituationImportance    SET activeClassSize = NULL    WHERE (activeClassSize = 0)

/* Compute the class and situation ratios. */

```

```

UPDATE #SituationImportance
    SET classRatio = (1.0 / classSize),
        activeClassRatio = (1.0 / activeClassSize),
        situationRatio = situationRuleCertaintySum / classRuleCertaintySum

/* Compute the situation distances */
UPDATE #SituationImportance
    SET situationDistance = dbo.UFN_Abs(situationRatio - classRatio),
        activeSituationDistance = dbo.UFN_Abs(situationRatio - activeClassRatio),
        countDistance = dbo.UFN_Abs(countRatio - activeClassRatio)

/* Compute the average class distance */
UPDATE #SituationImportance
    SET averageClassDistance =
    (
        SELECT AVG(situationDistance)
        FROM #SituationImportance AS Nested
        WHERE Nested.id_situationClass = #SituationImportance.id_situationClass
    )

/* Compute the maximum class distance */
UPDATE #SituationImportance
    SET maximumClassDistance =
    (
        (
            1.0 - classRatio
            +
            (
                (
                    classSize
                    -
                    1.0
                )
                *
                classRatio
            )
        )
        /
        classSize
    ),
    maximumActiveClassDistance =
    (
        (
            1.0 - classRatio
            +
            (
                (
                    activeClassSize
                    -
                    1.0
                )
                *
                activeClassRatio
            )
        )
        /
        activeClassSize
    )

UPDATE #SituationImportance
    SET situationImportance = (averageClassDistance / maximumActiveClassDistance)

CREATE TABLE #Result (id_situationClass bigint, id_situation bigint, situationImportance float)
INSERT INTO #Result (id_situationClass, id_situation, situationImportance)
    SELECT id_situationClass, id_situation, situationImportance FROM #SituationImportance

SELECT id_situationClass, id_situation, situationImportance FROM #Result

```





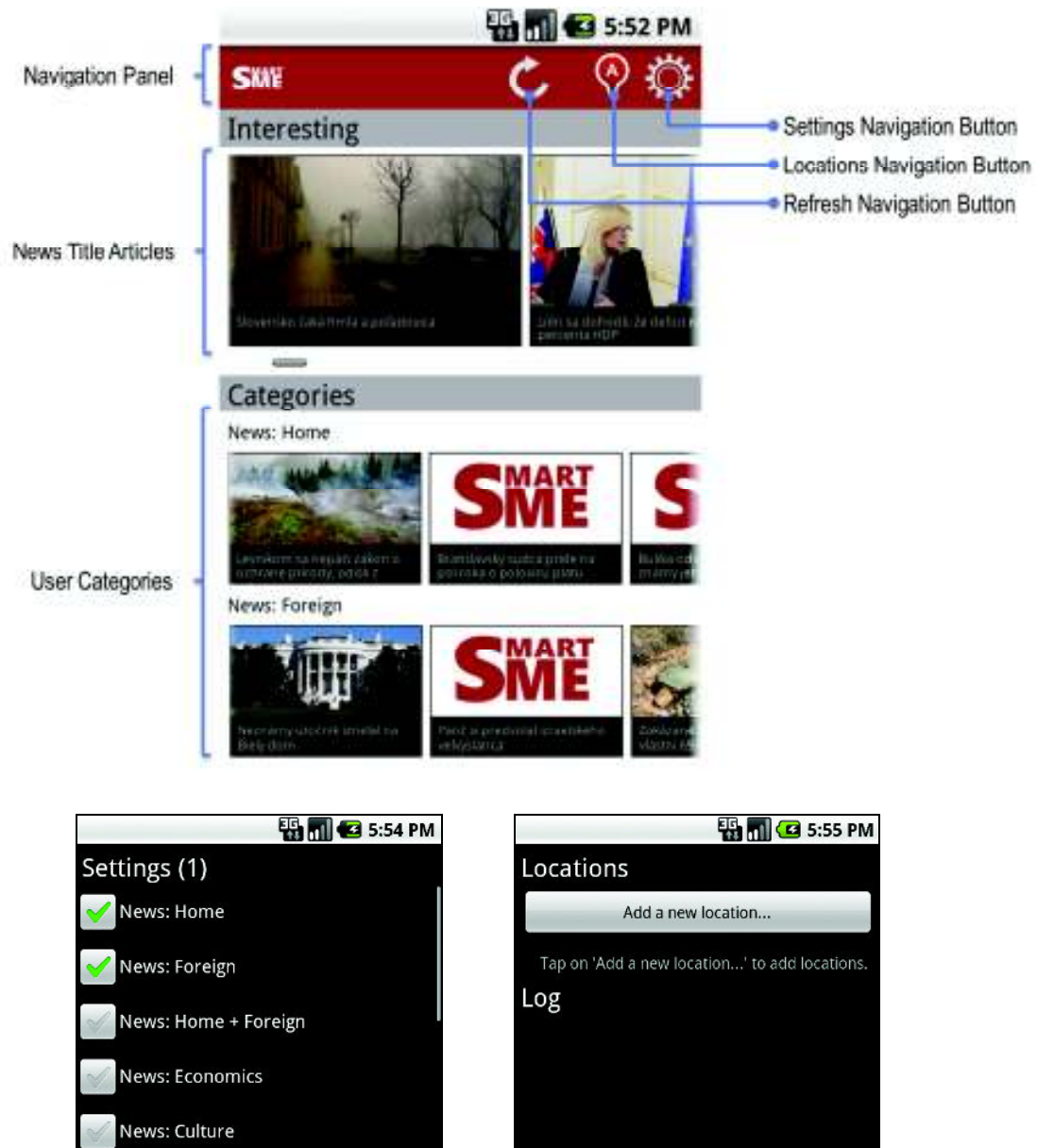
# Appendix B:

Reference Manual



# Mobile News Application

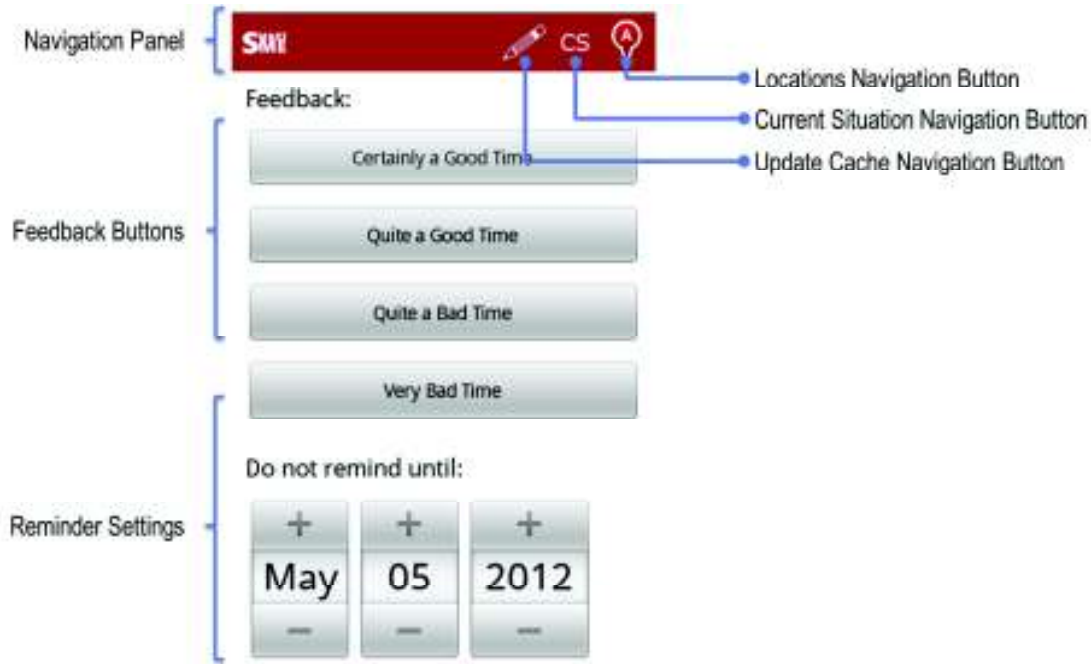
## Full Version



This is the original full version client for Android mobile devices. It logs situation observations and defines rules according to identified action indicator observations for the internet news domain. This original version also uses the content services that periodically process RSS feeds from sme.sk and process articles included. In addition to that the content services also allow setting and

manipulation of user preferences regarding the news content type (article categories).

## Experiment Version



The experiment version was created specifically for the purposes of our live experiment to be able to retrieve more reliable feedback in a short period we had allotted to it.

## Recommendation Service API

The service API serves as a front-end to our action recommendation engine. It provides services for defining users, situations, actions as well as logging observations and retrieving recommendations.

URI	Description
[API]/Users	<b>Retrieve all users</b>
[API]/Users/Details	<b>Retrieve details of a given user</b>
[API]/Users/Create	<b>Create a new user</b>
[API]/Parameters	<b>Retrieve all method's parameters</b>
[API]/Parameters/Set	<b>Set value of a given parameter</b>
[API]/Situations/Classes	<b>Retrieve all defined situation classes</b>
[API]/Situations/ClassDetails	<b>Retrieve details about a given situation class</b>

[API]/Situations/ClassSituations	Retrieve all situations of a given class (and user)
[API]/Situations/SituationDetails	Retrieve details about a given situation
[API]/Situations/DefineSituation	Define a new (user) situation
[API]/Situations/Observations	Retrieve all situation observations of a given user (for a given time)
[API]/Situations/ObservationCount	Retrieve an approximate number representing how many times a given situation for a given user in a given time frame has been observed
[API]/Situations/LogObservation	Logs a situation observation of a given situation, user, time, certainty and descend rate
[API]/Actions/Actions	Retrieve all defined actions
[API]/Actions/ActionDetails	Retrieve details about a given action
[API]/Actions/Observations	Retrieve all observations of a given user
[API]/Actions/ObservationDetails	Retrieve details about the given observation
[API]/Actions/LogObservation	Logs an action indicator observation of a given action, time, user, certainty and base descend rate. Also schedules rule definition for this observation.
[API]/Actions/Recommendations	Retrieve recommendations for a given user and time
[API]/Actions/SituationImportance	Retrieve situation importance for a given user and time



# Appendix C:

Paper Submitted to RecSys 2012

*(Based on IIT.SRC Paper)*





# Recommendation of Right Moment for Action Suggestion in a Recommender Based on Situation Rules

Anton Benčíč

Institute of Informatics and Software Engineering  
Faculty of Informatics and Information  
Technologies, Slovak University of Technology  
Ilkovičova 3, 842 16 Bratislava, Slovakia  
bencican@live.com

Mária Bieliková

Institute of Informatics and Software Engineering  
Faculty of Informatics and Information  
Technologies, Slovak University of Technology  
Ilkovičova 3, 842 16 Bratislava, Slovakia  
bielik@fiit.stuba.sk

## ABSTRACT

Nowadays we can see a new era of mobile computing spring up. Mobile devices more often than not provide incomparably more relevant information and context about their users than was ever available on desktops or within classic web browsing. With this a new branch of research for autonomous software is forming. The aim is to recognize usage situations and based on those let an application decide on and perform an action autonomously. In this paper we describe our novel method for learning users' situation preferences to recommend right moments for recommending actions independently of particular recommender be it news, songs or microblog recommender. User preferences are described with situations the users encounter throughout the time and rules that are based on either implicit or explicit feedback from the users. The focus of this paper is on the recommendation method for the right moment of action suggestion itself alongside which we also introduce a few usage scenarios, discuss on characteristics and limits of our method and present experiments that evaluate on its performance in various scenarios.

## Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – *clustering, information filtering, relevance feedback, selection process*, I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods

## General Terms

Algorithms, Experimentation, Design, Human Factors.

## Keywords

Action Recommendation, Situation Model, Action Model, Situation Rule, Machine Learning.

## 1. INTRODUCTION AND RELATED WORK

We live in a world where the pervasive, omnipresent and autonomous computing is already present and expanding in many different areas of our lives. Recommendations can be also considered one such area. Today however in most cases the user has to initiate the interaction by stating her intentions. Other

approach employs the concept of notifications, i.e. the interaction is initiated by the application. However, most of the time these notifications are not presented to the users in a sophisticated way, especially in the right moment, but rather naively, only considering the content and not the user's situation and preferences that can be estimated based on her activity in a particular context.

In this paper we propose a novel approach to action recommendation based on situation rules. We refer to an action as to anything an end-user service or application can perform on its own. In autonomous news recommendation for example pushing news to the user's device can be considered a typical action. Our method is designed to support autonomous decision making in context-aware services and applications that need to make decisions on actions without a user initiation. For example, a news service that decides on the type of presented content like the one presented [16] can be extended to identify whether it is the right time to push some news content to the user. This would even further maximize the chances that the user responds positively and that an overall satisfaction is achieved. A trivial example of such news service that attempts to identify suitable moments is presented in [17]. An example from a different domain is an entertainment service that can decide when it is the right time to open the store and propose some music albums, games or movies.

The reason we need a rather sophisticated method is that clues for the decisions are often indirect and almost always user dependent, so we cannot tell or decide directly. A simple example would be that of a travelling user. We do not know directly if we should or should not push some news right now, because the user may or may not be busy while traveling. She may be using public transport, or driving her own car, which is nearly impossible to sense on a device. Moreover she may be using the public transport on sunny days and taking her car while there are showers outside for instance. That is why there are usually no simple decisions and why such end-user services and applications have to personalize their approach to every user.

There are a couple of methods already exploiting rules for automated decision making and performance. One such example is presented in [1]. The main problem with existing approaches is that they require their rule base to be set up beforehand by an expert, often being the user herself. Moreover these rules serve as explicit definitions of actions for specific situations making them useful only for simplistic and straightforward scenarios. Such example service is presented in [2] and [3]. In [2] the authors present a method that automatically switches sound profiles on the user's mobile device according to the sensed situation. In [3] the authors present a method for recommending leisure time activities

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference '10, Month 1–2, 2010, City, State, Country.

Copyright 2010 ACM 1-58113-000-0/00/0010...\$10.00.

based on what time of week it is and what venues are close to the reported GPS position by combining different recommendation models using pre-built rules.

This kind of straightforward approach is mainly suitable for context-aware services and applications that need no personalization and the actions performed by them depend purely on the contextual information. Classic examples of such applications may be adaptive mobile guides presented in [4] or vehicle interfaces that decide how and when to present information based on the driving situation at hand [5].

The novelty of our method is in providing means to automatically create rule base using contextual information and user's feedback and suggest actions based on it. The sophisticated combination of certainty factors, context information and rules makes our model well suited for discovering relationship among situations and actions even in scenarios where a human-defined rule base is impossible due to its potential complexity.

Our method is based on symbols which makes it applicable in various scenarios and usable within a range of domains and applications. The only requirement for an end-user service or application is to identify situation classes that might be important for autonomous decision making on actions in the particular domain and our method is then capable of identifying the particular situations that are suitable for individual users.

The paper is structured as follows. In Section 2 situation model which represents the user's environment through a set of symbols is described. Action model that consists of a set of automatically generated rules is described in Section 3. Our novel method for action suggestion is presented in Section 4. Section 5 is devoted to evaluation of proposed approach. We describe simulations that demonstrate basic and some advanced characteristics of our rule-based action recommendation method. The paper is concluded with discussion and conclusions.

## 2. SITUATION MODEL

Symbolic situation representation allows for situation models that consist of relatively simple strings (symbols), but allow for representation of a variety of situation classes, from the most simple to complex ones. Because of its flexibility we opted to represent the user's environment through a set of symbols that are later basis for creating rules.

Each symbol represents a particular atomic situation and consists of two parts, where the first part represents the situation class and the second part represents a particular situation within that class. An example may be situation class *Weather* with value *Clear* that would go as follows:

$$\text{Weather : Clear} \quad (1)$$

Every *user-situation* instance has a certainty value associated with it that represents how certain are we that the user is in such situation at a particular time. The values are in  $(0.0, 1.0>$  range, where 0.0 would mean that it is absolutely uncertain about the situation's occurrence and 1.0 means that we are absolutely sure about the situation's occurrence. This gives us more flexibility by allowing to have more than situation at a time from the same class, each with its own assigned certainty. This kind of fuzzyfication is very handy for symbol-based models, because it allows us to represent uncertainty as well as position in a segmented continuous interval. For example at 6 AM we can say that it is morning, but at 10 AM it is starting to be lunch time, or noon as well and the concept of parallel situations with assigned certainty allows us to represent it more correctly.

Situations are fed into our method for action suggestion at arbitrary times that is dependent mostly on the target client service or application and the conditions, like internet connectivity, service availability or battery state. The situations that are fed represent the user's environment state at a particular time and the certainty attached to them at that time should not be the same sometime later (considering another situation update of the same class has not been delivered yet). That is why we embedded a concept of *time sensitivity* that decreases situations' certainty values as the time passes and it does it using the following formula:

$$CF_t = \frac{CF_b}{(1+r)^t} \quad (2)$$

where  $CF_t$  is the resulting certainty factor at a time  $t$ ,  $CF_b$  is the base certainty initially assigned to a situation,  $r$  is the rate of cease that controls how fast is the situation update losing its certainty (see Figure 1) and  $t$  is the time in hours that has elapsed since the situation was registered.

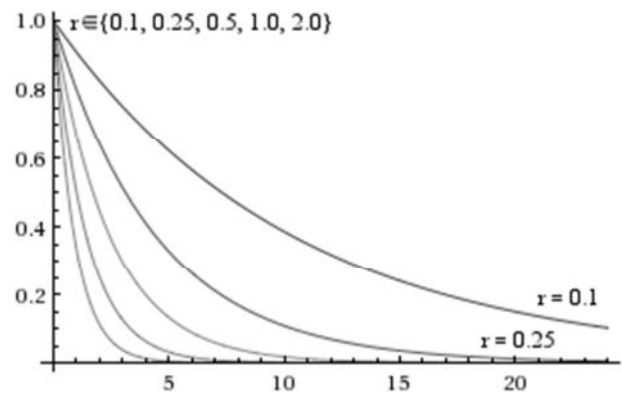


Figure 1. Certainty cease based on descend rate value.

One intrinsic advantage of such representation of user's situation is that the symbols bear no meaning with them thus eliminating privacy concerns that we might otherwise experience with some users. This allows for the recommendation engine to be easily deployed within any web service to allow for possible future collaborative models, where rules can be transferred among users based on their similarity to further speed-up the training process.

## 3. ACTION MODEL

The action model consists of a set of rules that are automatically generated based on the user's performance that is her implicit and explicit feedback. The first part of any particular rule is

- a set of antecedents that define in which situation the particular rule applies.

Every antecedent has a certainty value that represents the certainty of that atomic situation at a time that the rule was defined. Second part of every rule is

- its consequence (or action) that is a symbol similar to the situation symbols (section 2). The only difference is that conclusion symbols are not defined in classes.

Every rule is also assigned a certainty value that defines what weight a rule holds. For example, considering a news recommendation service a user opening a news application is a strong clue that this may be a right situation for presenting news

in future, while a user not responding to a notification is only a weak clue of the opposite. In such case for

- the positive feedback we define a rule with conclusion like *ShowArticles* and with certainty closer to 1.0

while in case of

- the weak negative feedback we define a rule with conclusion like *DontShowArticles*, but with certainty closer to 0.0.

The rules similarly to the situations employ a *time sensitivity* principle where the rules lose their certainty over time. We cannot however proceed as simply as with the situations because in case of sparse updates we would only have a couple of eligible rules to base our calculation on. The rules use the same formula for reduction of their certainty as situation with a slight modification that considers the rate at which situations from their antecedent set have appeared since the rule was defined.

The final formula goes as follows:

$$CF_t = \frac{CF_b}{(1+r)^t} \quad (3)$$

where  $r$  as a rate of cease is no longer a simple number present with any rule, but it is derived from the rate at which situations from the rule antecedents set have appeared since the rule was defined.

The formula for finding  $r$  goes as follows:

$$r = r_b \cdot \left( \left( \frac{\sum CF_{sa_1}}{m \cdot CF_{a_1}} \right) \cdot \left( \frac{\sum CF_{sa_2}}{m \cdot CF_{a_2}} \right) \cdot \dots \cdot \left( \frac{\sum CF_{sa_n}}{m \cdot CF_{a_n}} \right) \right) \quad (4)$$

where  $r_b$  is the base cease rate associated with the rule (this is similar to situations),  $CF_{a_n}$  is certainty of the  $n$ -th antecedent in the particular rule,  $m$  is the base antecedent certainty cease multiplier that roughly defines how many situation updates it takes to completely cease the rule's certainty and  $\sum CF_{sa_n}$  is the sum of certainties of situation updates that match the situation of the  $n$ -th antecedent. The parameters that control the resulting cease rate are  $r_b$  and  $m$ . Their best values may differ by domain, specifically depend on how often a feedback is registered and are best to be found experimentally.

## 4. METHOD FOR ACTION SUGGESTION

Our method is based on a set of automatically generated rules that are based on a feedback received from any particular user. All rules contain antecedent part that describes the situation in which the rule was defined and conclusion part that describes what action a rule suggests when the situation corresponds. All the rules are used within the computation model that aggregates them and calculates the final recommendations. Sections 4.1 to 4.3 describe how the rules are generated and recommendations for action suggestion computed.

### 4.1 Rule Generation

We define rules upon a client service or application notifies of an action suggestion. In such case the most recent situation update of all situation classes is taken to form antecedents for the newly created rules and the suggested action will be their conclusion. As we already mentioned there can be more than one valid situation from any situation class. In such case more than one rule is created in a way that every rule has exactly one antecedent from every situation class and there are rules for any possible combination of situations.

Following example demonstrates what rules would be defined if the state would consist of five situations from three classes:

Situation  $\rightarrow$  C1: S1, C1: S2, C2: S1, C2: S2, C3: S1

Rules  $\rightarrow$ 

- if C1: S1 and C2: S1 and C3: S1 then A
- if C1: S2 and C2: S1 and C3: S1 then A
- if C1: S1 and C2: S2 and C3: S1 then A
- if C1: S2 and C2: S2 and C3: S1 then A

where the base suggestion certainty provided by the target service is distributed among the new rules based on certainty of their antecedents compared to the certainty of antecedents from all newly defined rules:

$$CF_r = CF_b \left( \frac{\sum CF_{a_r}}{\sum CF_a} \right) \quad (6)$$

where  $CF_r$  is the final rule certainty,  $CF_b$  is base certainty assigned by the client service or application,  $\sum CF_{a_r}$  is the sum of antecedent certainties from the particular rule and  $\sum CF_a$  is the sum of antecedent certainties from all newly created rules.

After all new potential rules are created it has to be decided which ones will be included in the user's rule base. All rules that have no existing match (both their antecedent set and conclusion matches one of the rules) are included in the rule base. If a rule has a match in the user's rule base it is only factored in if it has greater final calculated certainty than the rule that is already present in the model. In such case the old rule is retired from the rule base. The calculation for the final rule certainty goes as follows:

$$CF_f = (CF_{a_1} \cdot CF_{a_2} \cdot \dots \cdot CF_{a_n}) \cdot CF_r \quad (7)$$

where  $CF_f$  is the final computed certainty,  $CF_{a_n}$  is certainty of the  $n$ -th rule antecedent and  $CF_r$  is the certainty initially assigned to the rule.

## 4.2 Recommendation Calculation

The recommendations for actions suggestion are calculated from the rules for a particular user. These are however first adjusted using the user's situation model to decide on their final weight.

The recommendation calculation is a three-step process:

1. Compute the importance of situation classes for particular user situation.
2. Modify rule certainties based on their antecedent certainties, current user's situation and importance of situation classes.
3. Compute the final recommendations based on present rules with their modified certainties.

### 4.2.1 Computing the importance of situation classes

Not all of the situation classes are important for a particular user and therefore their full inclusion in the computation process degrades recommendation results. To avoid this we have to identify how important a situation class is for a particular user, therefore finding its importance. To achieve this we compare how much is situation distribution from any particular class similar to a possible uniform distribution.

The first part is calculating ratio for any situation in a situation class in possible uniform distribution of situations (situation class ratio):

$$r_{sc} = \frac{1}{|SC|} \quad (8)$$

where  $r_{sc}$  is the situation class ratio and  $|SC|$  is cardinality of the situation class.

Second part is calculating the real individual situation ratios by comparing the presence of a particular situation in the rule base to the presence of all situations from the situation class:

$$r_{sm} = \frac{\sum CF_{sm}}{\sum CF_{s_1} + \sum CF_{s_2} + \dots + \sum CF_{s_n}} \quad (9)$$

where  $r_{sm}$  is the computed situation ratio for situation  $m$ ,  $\sum CF_{sm}$  is the sum of certainties of rule antecedents with situation  $m$  and  $\sum CF_{s_1} + \sum CF_{s_2} + \dots + \sum CF_{s_n}$  is the sum of certainties of rule antecedents with any situation from the situation class that is being processed (class of situation  $m$ ). Calculation of the ratio and thus all subsequent calculations are performed only with situations that have presence in the user's rule base. Situations that do not are after computing the *situation class ratio* left out.

Third part is calculating the distance of each situation ratio from its situation class ratio:

$$d_{sm} = |r_{sc} - r_{sm}| \quad (10)$$

where  $d_{sm}$  is the computed distance and  $r_{sc}$ ,  $r_{sm}$  are results from the previous computations.

The fourth part is computing the final importance of a situation class by comparing the average distance of situations computed in the previous step to the maximum possible average distance:

$$I_{sc} = \frac{\left( \frac{d_{s_1} + d_{s_2} + \dots + d_{s_m}}{m} \right)}{\left( \frac{(1-r_{sc}) + (m-1)r_{sc}}{m} \right)} \quad (11)$$

where  $I_{sc}$  is the final computed situation class importance,  $\left( \frac{d_{s_1} + d_{s_2} + \dots + d_{s_m}}{m} \right)$  is the average distance of situations ratios from their class ratio and  $\left( \frac{(1-r_{sc}) + (m-1)r_{sc}}{m} \right)$  is the maximum possible average distance.

The final computed values range from 0.0 to 1.0, where exact 0.0 is achieved when all the situations are present in the user's rule base equally (exact uniform distribution) and exact 1.0 is achieved when only one situation from a class is present in the whole rule base.

#### 4.2.2 Modifying rule certainties

In this step the rule certainties are modified based on the match between certainties of their antecedents and corresponding current situation certainties with the situation class ratio  $I_{sc}$  also being factored in. The first part is modifying the base rule certainty with regards to the *time sensitivity principle* presented in section 3.

The second part is computing how much each rule antecedent influences the final rule certainty:

$$F_a = 1 - I_{sc} \cdot \left( 1 - \text{Min} \left( 1, \frac{CF_s}{CF_a} \right) \right) \quad (12)$$

where  $F_a$  is the final influence factor of an antecedent  $a$ ,  $I_{sc}$  is the computed situation class importance for the situation of antecedent  $a$ ,  $CF_a$  is the antecedent  $a$  certainty and  $CF_s$  is the certainty of current situation corresponding to the situation of antecedent  $a$ . The resulting factor ranges from 0.0 to 1.0 with 0.0 being achieved when the importance of a situation class is 1.0 (most important) and the situation from antecedent  $a$  is not present in the current situation. Factor of value 1.0 is achieved whenever a

situation class has 0.0 importance or there is an exact or higher match between the antecedent's certainty and certainty of the corresponding situation in the user's current situation. The minimum function is employed for cases when the certainty of antecedent is lower than the corresponding situation certainty.

The third part is factoring the rule antecedent factors into the final rule certainty:

$$CF_f = (F_{a_1} \cdot F_{a_2} \cdot \dots \cdot F_{a_n}) \cdot CF_r \quad (13)$$

where  $CF_f$  is the final rule certainty,  $F_{a_1}$ ,  $F_{a_2}$ ,  $F_{a_n}$  are factors of antecedents 1 to  $n$  and  $CF_r$  is the original rule certainty.

This model ensures that the more important are the situations that are present in the rule antecedent set, but missing from the current situation the more is the rule certainty decreased. On the other hand if a situation with low importance is missing, the rule certainty is decreased just slightly.

#### 4.2.3 Computing the final recommendations

Computing the final recommendations groups rules from the user's rule base with matching conclusion (action) and sums their final certainties according to the summation formula proposed by David McAllister's model of working with certainties:

$$CF = CF_a + CF_b(1 - CF_a) \quad (14)$$

where  $CF_a$  and  $CF_b$  are two positive certainty factors that range from 0.0 to 1.0. If the conditions are held, the formula ensures that the final computed value is also within the range from 0.0 to 1.0 by adding the second certainty factor reduced by the remaining certainty. The parameters in the equation are interchangeable. Also if there are more than two certainties to sum together the order in which they are processed is not important, thus any order gives the same result.

### 4.3 Discussion

We designed our method in a way that it can be used in any domain as both situation and action models are domain independent. One important concept in our method is the *time aspect* which allows for a change of the user's action model once her preferences change. The rate at which our method is able change the model depends on the value of the base antecedent certainty cease multiplier introduced in section 3.1. The lower the parameter is, the faster the model can change. However too low values can cause an adverse loss of rules even when there is no need for the model to change. Thus the most suitable value for this parameter needs to be found and set experimentally.

One important problem when working with loads of situation data or context is selection of the important context. Generally when automated methods work with much more information (be it context or anything else) than necessary, they tend to work incorrectly, because there are too many variables that do not matter and thus obscure what is important and invalidate the results.

Our method performs well in this case of information flood. To demonstrate it, consider the following rule base:

$$\begin{aligned} &\text{if } C1: S1 \text{ and } C2: S1 \text{ and } C3: S1 \dots CN: S1 \text{ then } A \\ &\text{if } C1: S1 \text{ and } C2: S2 \text{ and } C3: S2 \dots CN: S2 \text{ then } A \\ &\text{if } C1: S1 \text{ and } C2: S3 \text{ and } C3: S3 \dots CN: S3 \text{ then } A \\ &\text{if } C1: S1 \text{ and } C2: S4 \text{ and } C3: S4 \dots CN: S4 \text{ then } A \end{aligned} \quad (15)$$

where only  $C1:S1$  is the important situation and the rest of them is unimportant to the current user. If we were looking for exact rules then in case a completely new situation only with  $C1:S1$  held would arise, we would not know what to do. Our method however

considers all of the four present rules, because they all contain *CI:SI*. Their certainty is lowered (as explained in section 4.2) because only one of the antecedents matches the current situation, however a spike in the certainty for recommendation of A is still present and can be detected. This is further emphasized by the situation class importance that is able to identify randomness among the rule antecedents. An experiment for such case is presented in section 5.

It is possible create rules with negative certainties that would lower the final recommendation certainty for any particular action, however we advise using an opposite action as it is cleaner and gives more space for making decisions on the final action. If however negative rule certainties are used there are different summation equations to work with them:

$$CF = -(CF_a + CF_b(1 - CF_a)) \quad (16)$$

where both  $CF_a$  and  $CF_b$  are negative certainty factors ranging from -1.0 to 0.0.

$$CF = \frac{(CF_a + CF_b)}{(1 - \min(CF_a, CF_b))} \quad (17)$$

where both  $CF_a$  is positive certainty factor ranging from 0.0 to 1.0 and  $CF_b$  is negative certainty factor ranging from -1.0 to 0.0 or vice versa.

An important note to point out is that our recommendation method and models described in this paper are not sensitive to the level at which the rule certainties are set, but the end-user service or application just needs to be consistent and assign comparably higher certainties to feedback that is a stronger clue of an action suitability and lower certainties when observing weaker clues. Because of this there is also no specific certainty at which the end-user service or application should perform a specific action, but the best time is when the certainty reaches for a significant local maximum in its development through time or is in a significant incline.

## 5. EVALUATION

For evaluation on our method's performance we have implemented a simulation framework and a mobile news recommendation application. Using the simulation framework we have performed a series of simulations for refinement of our method's parameters, so as to be able to reliably identify situations for recommending specific actions.

The simulation framework works in two stages. The first stage is responsible for generating an environment (situations) of an imaginary user for a set period of time (i.e. one month).

The second stage is then responsible to simulate feedback based on the type of chosen user. This can be a user that prefers to read news during mornings, a user who only reads news on showery Monday evenings or a user whose preference changes over time. These are just a few examples and the simulations we have performed with different environments and types of users are presented in the following section.

### 5.1 Simulations

The purpose of simulations is to demonstrate basic and some advanced characteristics of our rule-based action recommendation method. To achieve this we advance from the simplest simulations and scale them up as we progress. The situations in the situation classes are represented through symbols that cluster real or discrete values (the purposes and approaches to raw context clustering are presented in [11] and [13]). For example the

*TimeOfDay* situation class contains twelve symbols that cluster 24 hours of a day into 2 hour segments. The current time is then presented as one to three symbols with certainty assigned to them. This certainty is assigned where in the or how far around any segment the current situation fits according to a normal distribution with span dependent on the cluster size.

The first simulation we ran was a simulation of environment with only one situation class and a simulation of a user whose preference was to read news in the morning from 7:00 AM to 11:00 AM (see Figure 2).

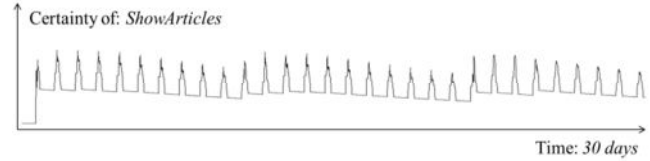


Figure 2. One-class, simple preference simulation results.

The environment of the first simulation contained one situation class (*TimeOfDay*) with the simulated user providing feedback in the morning which spanned over four hours, with the feedback being most probable at 9:00 AM and spreading with normal probability distribution to 7:00 AM and 11:00 AM. It is important to note however that when creating rules (providing feedback) we have always provided our method the same certainty. This corresponds with a real-world scenario where you do not adjust the feedback's certainty based on the situation's suitability because you do not know yet how suitable the situation is. The simulation ran over.

Another important point is that we have provided feedback in the first five days then continued five days without any feedback and so on. Also within the five day periods when we provided feedback by default were a few days without it. The reason is that we never exceeded over 80% probability for providing it. The results show clear peaks that suggest for example when it is the right time to push some news to the particular user's mobile device.

Our second simulation builds up on the first one with addition of one situation class with two situations (*Clear*, *Cloudy*) that represent current weather situation (see Figure 3).

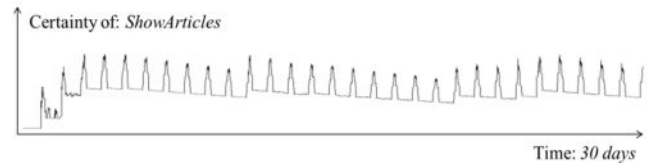


Figure 3. Two-class, simple preference.

The purpose was to evaluate on how is our method capable of identifying unimportant situation classes (described in section 4.2.1). This experiment shares other characteristics with the first one. Despite a little rough start when the model was adjusting itself to the simulated user the peaks can be clearly identified both in the beginning and throughout the whole simulation.

Our third simulation is a slight modification of the second simulation, where we exchanged the trivial weather situation class with class that identified days of week and thus contained seven different situations.

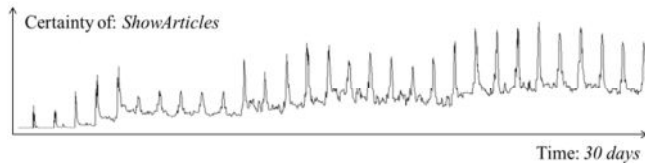
With scaling up we can see that our method is learning a little longer than in the previous case. On the other side peaks are still

clearly identifiable even within days six to ten where no feedback is provided at all (see Figure 4).



**Figure 4. Two-class, simple preference (extended).**

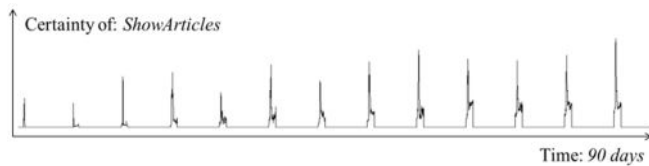
In our fourth experiment we scaled up number of followed classes to nine leaving other characteristics intact (see Figure 5).



**Figure 5. Nine-class, simple preference.**

In this simulation we had eight situation classes that were unimportant for the simulated user and only one that was important. The characteristics of the eight unimportant simulated situation classes varied too. Among them were contained classes that progressed steadily with slow, average and fast rate, situation classes that changed value fairly randomly with both small and large jumps, and a situation that showed almost no change at all throughout the whole simulation. The number and variety of unimportant situation classes employed in this simulation showed its traces in the overall appearance of the output certainty graph, but the peaks are still significant and thus reliably identifiable.

In our fifth simulation we have kept all the situation classes from our previous simulation but exchanged the simulated user for one whose preference is to read news only on Monday morning. Because there are only a few Monday mornings per month we have also extended the simulation span to 90 days. The purpose of this simulation is to examine characteristics of our method when two situations from two different classes are important when together (see Figure 6).



**Figure 6. Nine-class, composed preference.**

Because all the rules contain situation of *Monday* with 1.0 certainty it is considered to be most important. This is the reason why the certainty stays up through all Mondays. The other important class is more spread, because not only *Morning* situation is present within the rules, but there are also *EarlyMorning* and *LateMorning* antecedents within the rules. This lowers the overall class importance to somewhere around 0.7 – 0.8 and therefore even when it is Monday evening the rule certainties do not drop to zero as the *TimeOfDay* class does not have the top 1.0 importance and the *DayOfWeek* class has.

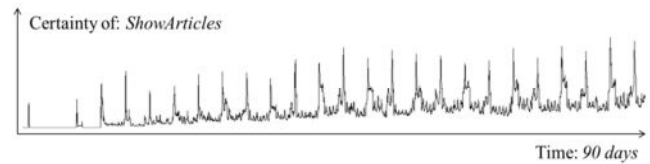
Remember that the purpose of our method is to provide significant peaks that are easily identifiable which is still held even in this borderline case. If we modified the simulated user preferences a

bit making her willing to read news on Mondays both in the morning and in the evening, we would see another significant spike at the end of every Monday evening which is exactly what we need.

Besides the ability to accommodate to a user with two interlinked important situations from two different classes the results of this experiment also show our method's ability to handle rule retirement correctly. A simple time aspect model like the one used with situations would be impossible to set up with rules. It would either count with frequent feedback or with sparse feedback.

In case the model would count with frequent feedback and thus rule creation the rule certainties in this would be degraded before the next week comes. If the model would count with sparse feedback it would not allow for preference change because the rules would be held in the user's rule base for too long. As explained in section 3 our method considers the rate at which situations from rule antecedents appear over time and bases rule retirement on that.

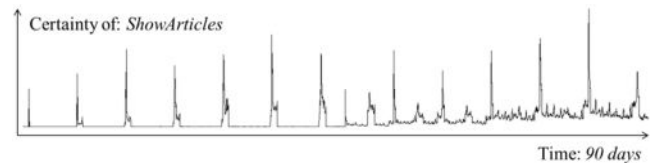
Our sixth experiment simulation scales the fifth one and studies a case of a simulated user for whom there are two pairs of important situations. The simulated user prefers to read news on Monday morning and Friday evening (see Figure 7).



**Figure 7. Nine-class, multiple composed preferences.**

This experiment simulation was again run throughout a period of 90 days. Even though this time we have two different pairs of important situations within nine different classes our method is still able to reliably identify the important ones after only a few of their occurrences. The gap in the first week's Friday is caused by our simulated user not providing feedback on that day. There are a couple other days where no feedback is provided, however rules from the past weeks are still in force and thus the peaks are present in the final results.

Our last experiment simulation tests out the rule retirement model by simulating a user who prefers to read news on Monday morning, but after one and a half month his preferences change to Friday evening (see Figure 8).



**Figure 8. Nine-class, change of preferences.**

This last experiment also spans throughout 90 days and its results clearly demonstrate our method's capabilities to change when users change their behavior. Throughout the first one and a half month the simulated user gives positive feedback on Monday morning and our method observes and learns this properly as is the case with our fifth simulation. After roughly one and a half month the user's preferences suddenly change to Friday evening which is represented in the graph by the first spike right after for Monday

morning is around the same size for it has been lowered by lack of feedback on that day, but mainly by addition of new rules for another day for the first time. This as described in section 4 lowers the importance of the *DayOfWeek* situation class. For the rest of the time allotted for the simulation the rules for Monday morning lose their certainty according to the model described in section 3 and rules for Friday evening are becoming stronger. Because the rules for Monday have little to no influence in the upcoming weeks the importance of the *DayOfWeek* situation class is raised again as well.

To further evaluate on performance of our method within the simulations we have designed a very simplistic method for identification of significant spikes within a volatile series of data like the experiment results are. The first step is identifying the largest possible ascend or descend (change) in recommendation certainty value. The value of change is between recommendation certainties at time  $t_a$  and  $t_b$  is defined as follows:

$$\Delta CF = \frac{\sum_{n=a}^b (CF_{t_{n+1}} - CF_t)}{t_b - t_a} \quad (18)$$

where  $\Delta CF$  is the certainty change between two points in time,  $\sum_{n=a}^b (CF_{t_{n+1}} - CF_t)$  is the sum of certainty differences between sample times within the period from  $t_a$  to  $t_b$  and  $t_b - t_a$  is time difference between the two points in time.

Let us consider we are deciding whether the certainty at time  $t_c$  can be considered a significant spike and thus the time identified as suitable for performing the corresponding action. To find out we have to first find for every point in time between  $t_0$  and  $t_c$  the most significant possible change. Note that if as we go backwards from any point  $t_b$  (moving  $t_a$  towards  $t_0$ ) the time span is raising as well and therefore finding minimum (maximum) value in interval  $<t_0, t_b>$  does not ensure that it is our  $t_a$  that maximizes (18) for  $t_b$ . Also note that the data is not continuous but rather sampled with some interval which means that the requirement for finding the largest possible change for every point in time is reduced to finding it for all times at which we have certainty samples.

The second step is finding the *average of largest changes* ( $E$ ) for all points in time and *average of the  $k$  largest changes* ( $E_k$ ). For our evaluation we use  $k = 10$ . Then the threshold for deciding whether we identify the certainty at any particular time point as a significant spike is set as:

$$T = \frac{E + E_k}{2} \quad (19)$$

thus being the average of  $E$  and  $E_k$ . Note that the threshold value is compared to the maximum change of certainty for any point in time as defined in (18), because our aim is to identify significant spikes rather than significant values.

Using the following significant spike identification model we have evaluated on how well our method performed in the simulations. We consider all the suitable situations of a user within a simulation to be times where we expect a spike to be identified and all other times where we expect it not to be identified. This goes for situations that are considered suitable by the imaginary user even though no feedback was provided at that particular time. The precision, recall and F-measure values for all seven simulations are presented in Table 1.

The statistics even with our simple method for spike identification show very good results where most of the deteriorations are caused because of lack of short span and overall lack of data in the beginning. Lowering the  $k$  parameter when there is not yet sufficient data improves the results as well as leaving out the first

day in 30 day simulations and first week in 90 day simulations. The values of precision, recall and F-measure in brackets represent computed values when the first day (week) are left out of the computation.

**Table 1. Statistical evaluation of experiment simulations.**

Simulation	Precision	Recall	F-measure
Simulation 1	0.88 (1.00)	0.97 (1.00)	0.92 (1.00)
Simulation 2	0.91 (1.00)	0.97 (1.00)	0.94 (1.00)
Simulation 3	0.94 (0.97)	0.97 (1.00)	0.95 (0.98)
Simulation 4	0.94 (1.00)	1.00 (1.00)	0.97 (1.00)
Simulation 5	0.98 (1.00)	1.00 (1.00)	0.99 (1.00)
Simulation 6	0.98 (1.00)	0.97 (1.00)	0.97 (1.00)
Simulation 7	0.94 (0.96)	0.87 (0.92)	0.90 (0.94)

## 5.2 Discussion

The purpose of the experiment simulations was to testify that our rule-based action recommendation method and especially its mathematical models that underline it are correct. We have chosen to perform simulations with their advancing complexity to discover our method characteristics as the data complexity scales up in different directions.

The experiments have shown that our rule time sensitivity model works much more reliably than a simplistic model like the one used with situations work here. In multiple scenarios we have seen that the certainty does not drop to zero even when it is clear that it is not the right situation for a particular action. This is okay, because as we already stated we are interested in significant spikes and possibly inclines of the resulting recommendation certainty.

Even though we have seen our method perform well even when scaled up, the wobbliness is present more significantly when more unimportant situation classes are employed. This is a well-known problem in recommendation where adding more inputs does not always produce better results and it is thus important to identify and use only the relevant context [15]. But again due to the underlying mathematical models of our method we are able to deal with reasonable amount of clutter very well. On the other side to make our method perform its best it is up on the target end-user service or application to define and use the classes that are important in the domain. For example in case of a real push news service we would employ situation classes that tell us how long ago the last session was, how many articles the user has read during it or how long it lasted.

## 6. CONCLUSIONS

Our method aims at providing a simple and flexible means for context-aware services and applications that need to provide specific personalized content or perform specific personalized actions autonomously in specific situations. The main strengths of our method is its use of abstract symbols and an intelligent time-sensitivity model.

The use of abstract symbols makes it domain-independent and simple to use while our time sensitivity model makes it possible to recognize and accommodate in user behavior or preference changes. An example of this is a user who travels during the spring and summer season with public transport making it an ideal

situation for reading some news but during autumn and winter commutes by driving her car making it impossible to read any news during that time.

To evaluate on our method's performance we have created a simulation framework that we used to run both simple and more complex simulations to ensure the correctness of our method's underlying mathematical models. Besides that we implemented our method into mobile news application and, using the work in [6] and [7] for news content recommendation, plan a long term live experiment.

There are a couple of possible enhancements or modifications to our method as well. The first one is including a support for collaborative models, where rules could be transferred and shared among similar users just as content is recommended in this collaborative way [8]. This may significantly counter the known cold-start problem for new users, after a sufficient user-base is already active in the target service.

Another extension is to include a base importance factor for situation classes that would help us identify important times for different actions faster and more reliably even with a plenty of situation classes that are not generally important in a particular domain, but may be for some users and thus have to be present.

A possible extension is inclusion of lightweight ontologies or similar linking schemes as presented in [9], [10] and [12]. This would allow for rule derivation using similarity or parental links among concepts, in this case situations and actions which could also lead to quicker learning and more reliable results. On the other hand such extension would significantly complicate the overall simple method concepts and thus make it less accessible in development of context-aware end-user services or applications. A trade-off can be achieved by only exploiting hierarchical dependencies as presented in [14] as such hierarchy among classes and their situations can be easily employed using just our symbolic representation of situations.

We have designed our method to work in a variety of domains and scenarios. Even though we are still not past the live experiments the simulation results suggest that our method can be used for example for identification of suitable times for content presentation, for automatic sound profile switching, for webpage pre-loading or in any other scenario when there is a need for automated actions performed by an end-user service or application. We believe that our method and frameworks based on it will due to their simplicity encourage more services and applications to become context-aware and more intelligent.

## 7. REFERENCES

- [1] Korpipää, P., Häkkinä, et al.: Utilising Context Ontology in Mobile Device Application Personalization. In: *MUM*, Maryland, (2004), pp.133-140.
- [2] Ala-Siuru, P., Tapani, R.: Understanding and recognizing usage situations using context data available in mobile phones. In: *ubiPCMM'06: Proc. of the 2nd International Workshop on Personalized Context Modeling and Management for UbiComp Applications*, (2006).
- [3] Roberts, M., Ducheneaut, N., et al.: Scalable Architecture for Context-Aware Activity-Detecting Mobile Recommendation Systems. In: *WoWMoM 2008: Proc. of the 9th IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*. IEEE, (2008), pp.1-6.
- [4] Krüger, A., Baus, J., et al.: Adaptive Mobile Guides. In: *The Adaptive Web: Methods and Strategies of Web Personalization*, LNCS 4321. Springer, Berlin, (2007), pp. 521-549.
- [5] Bellotti, F., Gloria, A., et al.: COMUNICAR: designing a multimedia, context-aware human-machine interface for cars. In: *Cognition, Technology & Work*, London, (2005), pp.36-45.
- [6] Suchal, J., Návrát, P.: Full Text Search Engine as Scalable k-Nearest Neighbor Recommendation System. In: *IFIP Advances in ICT. Artificial Intelligence in Theory and Practice*, (2010), pp. 165-173.
- [7] Bielíková, M., Kompan, M., et al.: Effective Hierarchical Vector-Based News Representation for Personalized Recommendation. In: *Computer Science and Information Systems*, (2012), pp. 303-322.
- [8] Soller, A.: Adaptive Support for Distributed Collaboration. In: *The Adaptive Web: Methods and Strategies of Web Personalization*, LNCS 4321. Springer, Berlin, (2007), pp. 573-595.
- [9] Costa, A., Guizzardi, R., et al.: CORES: Context-aware, Ontology-based Recommender system for Service recommendation. In: *CAiSE'07: Proc. of the 19th International Conference on Advanced Information Systems Engineering*, (2007).
- [10] Liu, L., Lecue, F., et al.: Using Context Similarity for Service Recommendation. In: *ICSC'10: Proc. of the 4th IEEE International Conference on Semantic Computing*, Carnegie Mellon University, Pittsburgh, (2010), pp. 277-284.
- [11] Zhang, Y., Cai, S., et al.: A Study on the Method of Mobile Content Recommendation Based on Situations. In: *ISCID'10: Proc. of the 3rd International Symposium on Computational Intelligence and Design*, Hangzhou, (2010), pp. 23-26.
- [12] Xiao, H., Zou, Y., et al.: An Approach for Context-aware Service Discovery and Recommendation. In: *ICWS'10: Proc. of the 17th IEEE International Conference on Web Services*, Miami, (2010), pp. 163-170.
- [13] Shin, D., Lee, et al.: Context-Aware Recommendation by Aggregating User Context. In: *CEC'09: Proc. of the 11th IEEE Congress on Evolutionary Computing*, Vienna, (2009), pp. 423-430.
- [14] Biegel, G., Cahill, V.: A Framework for Developing Mobile, Context-aware Applications. In: *PerCom'04: Proc. of the 2nd IEEE International Conference on Pervasive Computing and Communications*, Orlando, Florida, (2004), pp. 361-365.
- [15] Yap, G.-E., Tan, A.-H., et al.: Discovering and Exploiting Causal Dependencies for Robust Mobile Context-Aware Recommenders. In: *IEEE Transactions on Knowledge and Data*, vol. 19, (2007), pp. 977-992.
- [16] Cantador, I., Castells, P.: Semantic Contextualisation in a News Recommender System. In: *RecSys'09: Proc. of the 3rd ACM Conf. on Recommender systems*, New York, (2009)
- [17] Yeung, K., Yang, Y.: A Proactive Personalized Mobile News Recommendation System. In: *DeSE'10: Proc. of the 3rd International Conference on Developments in eSystems Engineering*, London, (2010), pp. 207-212.



# Appendix D:

Journal Article Draft



# Timing is of Essence

## 1 Introduction

We live in a world where pervasive, ubiquitous and mobile computing is all around us and gets deeper into our lives with every passed day. The problem however is that most of this technology is not very user-sensitive, because the primary goals often do not include adaptation to user needs and preferences. For example most proactive content provision services like news applications push notifications to the user's device only based on the content and more often than not do not consider such important aspect as the user's current context. This often ignored and overlooked context can however not only provide clues on what type of content to provide so as to higher the overall satisfaction, but can also be a clue as to whether it is a good time at all to push any news which we believe is an important aspect that can even further raise overall satisfaction.

There are plenty of other examples when the right timing of actions can substantially raise the quality of end-user services and applications. For example imagine an email or SMS notification sounds that are not played at the time you receive the mail or text, but rather when your device senses that you are around. Or a more sophisticated service that recognizes your application usage patterns and adjusts main screen application shortcuts according to your preferences in the sensed situation. These are just two examples of the wide possibilities where pervasive computing can appear a little bit more intelligent.

Our aim is on scenarios from the last example. Specifically we aim at aiding methods and end-user services and applications that need to make autonomous decisions based on the sensed situation when there is no clear mapping between the two due to differences between individuals and circumstances that are not known upfront.

## 2 Information Recommendation Process

When we think about who initiates a recommendation there are in general two types of recommenders. The first is a group of "on-demand" recommenders and

the second is a group of proactive recommenders that are of our primary focus. Proactive recommendation consists of three aspects:

- Time of recommendation
- Content to recommend
- Presentation of the recommended content

All of these three aspects form a single coherent unit where one aspect depends on all other aspects and failure to understand this dependence may lead to substantial problems with the recommendation process and its results. To fully understand the three aspects in proactive recommendation we first have to introduce the concepts of *situation* and *context* and the way we understand them in our work.

## 2.1 Context and Situations

One of the definitions in [1] states that situation is a position of a person with regard to circumstances. In our view the circumstances characterize a state of a user or an environment she is in. This can be for example what the place, time or weather is for a particular user or even what is she doing or who is around. Regarding context there are many definitions of what it actually is and how we can structure it. The most suitable view from our perspective is the content view because it can be used for categorization of context information according to its use within the three aspects of the information recommendation process. The content view on context distinguishes three context categories:

- Device context
- User context
- Background context

The device context includes such information as battery level, screen resolution, processing power or connection quality that describes the device and not the user. This information is mostly valuable for context-aware applications that focus more on the presentation aspect than on what to recommend (content) or when to recommend (time).

The user context and background context on the other hand describe the situation that the user is currently in and thus are of our primary concern. They are sometimes listed separately, but the difference between them comes down to what the context describes more. Because these nuances are not that important for the purposes of our project we further refer to them as to user and

background context or simply user context. Here are just a few examples of commonly used user context information:

- Date and time
- Location
- User's calendar
- User's multimedia gallery
- Social feed
- Stored documents
- Browsing history
- Current weather
- Nearby facilities
- Nearby people

## 2.2 Time Aspect in Proactive Recommendation

Proactive content recommendation services have to recognize when it is the right time to provide some content or a piece of information and act upon it. In the most trivial case these can be times explicitly defined by the designer. For a news service this may be a few minutes after the alarm goes off, a few minutes before lunchtime or in the evening after the lights in the room are turned off. An example of such naïve method is presented for example in [2] where authors introduce a method that rearranges icons on the user's mobile device based on what facility she is at (they included market, mall, airport and cafeteria). Many other examples of naïve methods come also from the domain of mobile guides presented thoroughly for example in [3] and an example of a semi-automated approach to performing actions based on user-defined rules is presented for example in [4]. The problem however is that such approaches are suitable only for simplistic scenarios where the situations to actions connections are simple and straightforward. When these however begin to be more complex such that even the users themselves cannot exactly and explicitly define the right situations the use of such methods becomes cumbersome, impractical and overall limited.

Contrary to the naïve methods are the adaptive ones. An example of such method is presented also in [5] where authors introduce a method that automatically switches sound profiles on user's mobile device according to the user's preferences and based on sensing devices that are around. Another example of adaptive method is presented in [6] where authors introduce a method for recommending leisure time activities initially based mostly on proximity of available facilities, but later on incorporating user's situational preferences as well.

## 3 Domain of Internet News

### 3.1 Existing Solutions

In scope of the internet news domain, which we target as our first domain for application of our method, a few methods already exist that not only focus on recommending content, but also exploit context information or even provide proactive news access. The first example is introduced in [7] where the authors present their method for proactive news recommendation based on the currently viewed web page. In [8] the authors present their News@hand concept where they use context information to distinguish between general users' interest categories based on the situations she is in.

Class diagrams in this section represent views on the important parts of the two interlinked models that are used for storage of situation classes, situations and their observations as well as storage for actions, indicator observations and rules that the recommendations are based on.

### 3.2 User Behaviour Observations

The authors in [9] present an experiment where they used Google News service to assess the behaviour of user while reading news, but especially from the perspective of their collaborative recommender and its recommendation effectiveness. Within the experiment authors discovered that preferences of users are composed of two groups. The first group contains genuine preferences that are usually derived from the user's profession or hobby, while the second group encompasses trend preferences that depend on current wide events as for example the Olympics, World Cup, elections and so on. These facts served them as basis for a new collaborative recommender they designed and experimentally deployed. The results of the experiment showed that a raise in recommendation quality not only raises volume of recommended article visits compared to all articles visited but also the frequency of visits of their users. On the other hand length and number of articles read per session stayed roughly the same.

We have also performed our own experiment with user behaviour patterns, but our attention was aimed at discovering trend characteristic of different article types. In the experiment we have analysed approximately 136 million visits where we identified three basic types of news articles:

- Short-term articles

- Medium-term articles
- Long-term articles

Short-term articles are articles that had most of the view counts in the first three days from the time they are published and then the interest drops to only a few visits a day. Also, in most cases of the short-term articles the most visits are during the first 24 hours. These are articles about events that happen, are discovered or announced unexpectedly.

Medium-term articles show irregular spikes and gaps in different days reflecting a development of a time-spread event. These are for example articles about the World Cup schedule or an election event result predictions.

Long-term articles on the other side show steady view count over a longer period of time. These are articles about time-independent topics such as yearly horoscopes, health or cooking.

## 4 Method for Action Recommendation Based on Situation Rules

With our method we aim at providing means to support autonomous situation-based decision within end-user services or applications. We did not design our method for any specific domain, but we instead designed a method that can be used in various fields. Even though location and time context information are most widely used within context-aware methods and services we still need specific context information in the specific domains to be able to capture all the possibly important parts of the user's context.

### 4.1 Situation Model

The aforementioned examples of domain specificity required us to design our recommendation method in such way that these specifics can be easily captured and incorporated. To achieve that we decided to build our method using symbolic representation for both user's context (situations) and actions performed by end-user services or applications that define these symbols based on what they need and consider relevant. Symbolic situation representation allows for situation models that consist of relatively simple strings (symbols), but allow for representation of a variety of situation classes, from the most simple to complex ones. Each symbol represents a particular situation and

consists of two parts and for example for a situation of clear weather the symbol could be:

Weather : Clear

where *Weather* represents situation class and *Clear* in this case a situation within. Instances of these situations also called situation observations then constitute a user's situation model in time. Every situation observation has in addition to identifying symbol three other components:

- Time – the time at which it was observed
- Certainty – the level of certainty with the observation
- Descend rate – the speed of descending the certainty through time

The time informs us about the time of the observation, the certainty expresses how certain is the end-user service or application about the occurrence of the situation and finally descend rate is used lowering certainties of observations as the time passes. The formula for adjusting the certainty in time is:

$$CF_t = \frac{CF_b}{(1 + r)^t}$$

where  $CF_t$  is the resulting certainty factor at a time  $t$ ,  $CF_b$  is the base certainty initially assigned to a situation,  $r$  is the descend rate that controls how fast is the situation update losing its certainty and  $t$  is the time in hours that has elapsed since the time of the observation.

The value of the descend rate for any situation observation is set by the end-user service or application as is the case with certainty values. This gives flexibility to have observations that retain their certainty over a longer period of time as well as observations that lose their certainty very fast.

## 4.2 Action Model

The action model consists of a set of rules that are automatically generated based on the feedback from the end-user service or application and further on the action indicators they identify. An action indicator in any specific domain is anything performed by a user that indicates an appropriate time for a particular action. For example in the domain of news recommendation a strong action indicator for recommending some news is a user actually reading some news. Another indicator may be a user playing a game on her mobile device, because it may indicate that she is possibly free to be presented some interesting news.



Every rule consists of two parts. The first part is a set of antecedents that define in which situation the particular rule applies. Every antecedent is defined by symbol and a certainty value. The symbol corresponds to a particular situation from the situation model and the certainty value represents the certainty of the observation at the time when the rule was created.

The second part of every rule is its consequence or action that the rule suggests using action suggestion symbols (i.e. *ShowNews*). An action suggestion symbol is a symbol similar to the situation symbols with the only difference being that action symbols are not defined in classes. Every rule is also assigned a certainty value that defines what weight a rule holds.

The rules similarly to the situations employ a time sensitivity principle where the rules lose their certainty over time. Rules use the same formula for reduction of their certainty as situation with a slight modification that considers the rate at which situations from their antecedent set have appeared since the rule was defined. The final formula goes as follows:

$$CF_t = \frac{CF_b}{(1+r)^t}$$

where  $r$  as a rate of cease is no longer a simple number present with any rule, but it is derived from the rate at which situations from the rule antecedents set have appeared since the rule was defined. The formula for finding  $r$  goes as follows:

$$r = r_b \cdot \left( \left( \frac{\sum CF_{s_{a_1}}}{m \cdot CF_{a_1}} \right) \cdot \left( \frac{\sum CF_{s_{a_2}}}{m \cdot CF_{a_2}} \right) \cdot \dots \cdot \left( \frac{\sum CF_{s_{a_n}}}{m \cdot CF_{a_n}} \right) \right)$$

where  $r_b$  is the base cease rate associated with the rule (this is similar to situations),  $CF_{a_n}$  is certainty of the n-th antecedent in the particular rule,  $m$  is the base antecedent certainty cease multiplier that roughly defines how many situation updates it takes to completely cease the rule's certainty and  $\sum CF_{s_{a_n}}$  is the sum of certainties of situation updates that match the situation of the n-th antecedent.

### 4.3 Rule Generation

Rule generation is based on action indicator observations, where with every observation there are one or more rules defined. Rule definition is performed in three steps. In the first step the rules are generated together with their antecedents in such way that every rule has one antecedent from every situation

class present in the current user's situation model and combinations of all situations among situation classes are covered. Following example demonstrates what rules would be defined in an imaginary scenario where the state would consist of five situations from three different classes:

$$Situation \rightarrow C1:S1, C1:S2, C2:S1, C2:S2, C3:S1$$

$$Rules \rightarrow \begin{array}{l} \text{if } C1:S1 \text{ and } C2:S1 \text{ and } C3:S1 \text{ then } A \\ \text{if } C1:S2 \text{ and } C2:S1 \text{ and } C3:S1 \text{ then } A \\ \text{if } C1:S1 \text{ and } C2:S2 \text{ and } C3:S1 \text{ then } A \\ \text{if } C1:S2 \text{ and } C2:S2 \text{ and } C3:S1 \text{ then } A \end{array}$$

where the base suggestion certainty provided by the target service is distributed among the new rules based on certainty of their antecedents compared to the sum of antecedent certainty from all newly defined rules:

$$CF_r = CF_b \left( \frac{\sum CF_{a_r}}{\sum CF_a} \right)$$

where  $CF_r$  is the final rule certainty,  $CF_b$  is base certainty assigned by the client service or application,  $\sum CF_{a_r}$  is the sum of antecedent certainties from the particular rule and  $\sum CF_a$  is the sum of antecedent certainties from all newly created rules.

After all new potential rules are defined they have to be included in the user's rule base. All rules that have no existing match (their antecedent set and conclusion do not match any of the existing rules) are included in the rule base as they were defined. If a rule however matches one of the old rules, the old rule is dismissed and the certainty of the newly defined rule is adjusted as follows:

$$CF_a = \max \left( CF_{ot}, CF_r + (1 - CF_r) \cdot \left( CF_{ot} \cdot \left( \frac{CF_{ob} - CF_{ot}}{CF_{ob}} \right)^r \right) \right)$$

where  $CF_a$  is the adjusted certainty,  $CF_r$  is the new rule's original certainty computed in the previous step,  $CF_{ob}$  is the base certainty of the old rule and  $CF_{ot}$  is the certainty of the old rule at the time the new one is being defined. The aim of this model is to ensure that the rules defined roughly around the same time are not aggregated too much, but on the other side to allow aggregation of large part of the remaining certainty from the old rule should it withstand the test of time.

## 4.4 Recommendation Calculation

The rules that are defined over time serve us as basis within the actual action recommendation process that consists of three steps:

1. Compute the importance of situation classes
2. Modify rule certainties
3. Compute the final recommendations

### 4.4.1 Computing the importance of situation classes

Not all of the situation classes are important for a particular user and therefore their full inclusion in the computation process degrades recommendation results. To avoid this we have to identify how important a situation class is for a particular user using the following formula:

$$I_{sc} = \frac{\left( \frac{d_{s_1} + d_{s_2} + \dots + d_{s_m}}{m} \right)}{\left( \frac{(1 - r_{sc}) + (m - 1) \cdot r_{sc}}{m} \right)}$$

where  $I_{sc}$  is the computed situation class importance,  $d_{s_1} + d_{s_2} + \dots + d_{s_m}$  is the sum of distances in the actual distribution from the imaginary uniform distribution of situations in the given class,  $m$  is the number of situations and therefore  $\left( \frac{d_{s_1} + d_{s_2} + \dots + d_{s_m}}{m} \right)$  is the average distance of situations from the imaginary uniform distribution and  $\left( \frac{(1 - r_{sc}) + (m - 1) \cdot r_{sc}}{m} \right)$  is the maximum possible average distance.

### 4.4.2 Modifying rule certainties

After we have the importance of individual situation classes ready for a particular user we can carry on with modifying the rule certainties.

The first part is modifying the base rule certainty with regards to the time sensitivity principle (see Section **Error! Reference source not found.**). This gives us rules that have their base certainty lowered based on how much the situations from their antecedents have appeared since they were defined.

The second part is computing how much each rule antecedent influences the final rule certainty based on the situation class importance and match between the rule antecedent and the current situation. The formula for computing antecedent influence goes as follows:

$$L_a = L_{a_B} + (1 - L_{a_B}) \cdot (1 - I_{SC})^{p-(p-1) \cdot L_{a_B}^{\frac{1}{q}}}$$

$$L_{a_B} = \min\left(1, \frac{CF_S}{CF_a}\right)$$

where  $L_a$  is the final influence factor of an antecedent  $a$ ,  $L_{a_B}$  is its base influence factor of antecedent  $a$  that represents the match between antecedent certainty and certainty of the corresponding situation in the current situation model ( $CF_a$  is the antecedent's  $a$  certainty and  $CF_S$  is the certainty of current situation corresponding to the situation of antecedent  $a$ ),  $I_{SC}$  is the computed situation class importance for the situation of the antecedent  $a$  and the parameters  $p$  and  $q$  ( $p, q \geq 1$ ) control the shape and the change in the shape of the curve that transforms the base influence according to the situation class importance.

The third part is factoring the rule antecedent factors into the final rule certainty which goes as follows:

$$CF_f = (F_{a_1} \cdot F_{a_2} \cdot \dots \cdot F_{a_n}) \cdot CF_r$$

where  $CF_f$  is the final rule certainty,  $F_{a_1}$ ,  $F_{a_2}$ ,  $F_{a_n}$  are the factors of antecedents 1 to  $n$  and  $CF_r$  is the original rule certainty.

#### 4.4.3 Computing the final recommendations

Computing the final recommendations groups rules from the user's rule base with matching conclusion (action) and sums their final certainties according to the summation formula proposed by David McAllister's model of working with certainties:

$$CF = CF_a + CF_b(1 - CF_a)$$

where  $CF_a$  and  $CF_b$  are two positive certainty factors that range from 0.0 to 1.0. If the conditions are held, the formula ensures that the final computed value is also within the range from 0.0 to 1.0 by adding the second certainty factor reduced by the remaining certainty. The parameters in the equation are interchangeable. Also if there are more than two certainties to sum together the order in which they are processed is not important, in other words any order gives the same result.

## 4.5 Method Characteristics and Discussion

We designed our method in a way that it can be used in any domain as both situation and action models are domain independent. One important concept in our method is the *time aspect* which allows for a change of the user's action model once her preferences change. The rate at which our method is able to change the model depends on the value of the base antecedent certainty cease multiplier. The lower the parameter is, the faster the model can change. However too low values can cause an adverse loss of rules even when there is no need for the model to change.

One important problem when working with loads of situation data or context is selection of the important context. Generally when automated methods work with much more information (be it context or anything else) than necessary, they tend to work incorrectly, because there are too many variables that do not matter and thus obscure what is important and invalidate the results. Our method performs well in this case of information flood. To demonstrate it, consider the following rule base:

*if C1:S1 and C2:S1 and C3:S1 ... CN:S1 then A*  
*if C1:S1 and C2:S2 and C3:S2 ... CN:S2 then A*  
*if C1:S1 and C2:S3 and C3:S3 ... CN:S3 then A*  
*if C1:S1 and C2:S4 and C3:S4 ... CN:S4 then A*

where only *C1:S1* is the important situation and the rest of them are unimportant to the current user. If we were looking for exact rules then in case a completely new situation only with *C1:S1* held would arise, we would not know what to do. Our method however considers all of the four present rules, because they all contain *C1:S1*. Their certainty is lowered (as explained in Section 4.4) because only one of the antecedents matches the current situation, however a spike in the certainty for recommending action A is still present and can be detected. This is further emphasized by the situation class importance that is able to identify randomness among the rule antecedents.

It is possible to create rules with negative certainties that would lower the final recommendation certainty for any particular action, however we advise using an opposite action as it is cleaner and gives more space for making decisions on the final action. If however negative rule certainties are used there are different summation equations to work with them:

$$CF = -(CF_a + CF_b \cdot (1 - CF_a))$$

where both  $CF_a$  and  $CF_b$  are negative certainty factors ranging from -1.0 to 0.0.

$$CF = \frac{(CF_a + CF_b)}{(1 - \text{Min}(CF_a, CF_b))}$$

where  $CF_a$  is positive certainty factor ranging from 0.0 to 1.0 and  $CF_b$  is negative certainty factor ranging from -1.0 to 0.0 or vice versa.

## 5 Method for Action Recommendation in the Domain of Internet News

One of the problems in proactive news recommendation is that most today end-user services and applications consider only the content when pushing news to a user. These services mostly push news notifications when a generally and broadly important piece of news is published with some of them also considering the user's interests. There is however little work done in trying to find the right situations to push news to a user in order to maximize the chances of her actually reading them in order to maximize her overall satisfaction. This is one of the reasons we opted for the news to be our target domain and finding the right moments to be our target problem.

### 5.1 Situations

There are a few situations that we can explicitly use to tell us to what extent the user may or may not be now interested in some news content. As these form the basis of actions we discuss them later on in this chapter. What we are now more interested in is the context information that we cannot explicitly define as being either a clue for or against pushing a news notification onto a user's device. The context information sources (situation classes) that we identified with the news domain in mind are:

- Location status
- Time of day
- Day of week
- Week of month
- Month of year
- Current weather
- Calendar events
- Alarms
- Session
- Display

The situations in the aforementioned situation classes are symbols that can exist in two categories. The first category contains global symbols that are produced

by any particular client application. This is for example symbol of a user *in motion*. The other category consists of user symbols that are created during client runtime and can therefore only be recognized by that specific client instance – thus are relevant only to a specific user that defined them. An example of such situation may be a user *in motion from workplace to home*. Note that there are many users that have their *workplace to home* situations, but none of these are from the reasoning perspective semantically equal. On the other hand two users that exhibit the basic *in motion* situation are equal considering this criterion.

## 5.2 Actions and Indicators

There are also situations that we can explicitly classify in terms of suitability for pushing news to some extent and therefore they represent our action indicators. We identified four indicators that we consider suitable in the context of a news recommendation service and three indicators that suggest the time may not be right for pushing news (see Table 1).

**Table 1.** Indicators of situation suitability.

Indicators of suitable situation	Indicators of unsuitable situation
The user is reading news in our application	The user dismisses our push notification
The user is engaged in an entertainment application	The user ignores our push notification
The user is engaged in a productivity application	The user is ignoring an incoming call
The user is in a call	

Reading news is for most people a kind of side activity. In most cases it is neither required by their job nor is it a dedicated hobby. Finding the right time to push some news is therefore essentially finding the right time when a user can be distracted with something that does not necessarily relate to the matter at hand, but is interesting to the user. Domains such as social network statuses and updates, movie release information, interesting images, quotes, jokes, videos and a other content that shares the characteristic of being a side activity can be therefore treated in a much similar way and the concepts presented in this chapter apply to them as well.

## 6 Evaluation of the Method for Action Recommendation

For evaluation of our method we have implemented a simulation framework and a mobile news recommendation application. The experiment was three-staged.

In the first stage we performed a series of simulations to set-up our method's parameters and ensure that the inner workings are correct and can reliably identify suitable situations for a set of simulated environments and the users within.

In the second stage we distributed our modified mobile application to 6 users to evaluate on the characteristics of situation and feedback data. Our aim here was to verify the preconditions of our simulations to see if the simulated environments and users were representative of those from the real world.

In the third stage we took the knowledge from the live experiment in stage two and designed additional simulations that modeled the real environment and user behavior even more closely to their real counterparts.

### 6.1 Base Simulations

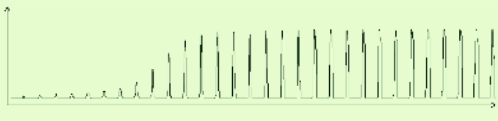
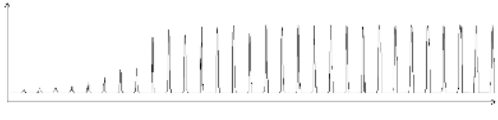

We have divided the base simulations into two groups. The first group represents simple scenarios and the second complex ones. All of the simulations we have performed considered only rules defined at least 12 hours before the time for which the recommendations were being computed in order to avoid computing recommendations for situations based on rules that were defined exactly in those particular situations.

#### 6.1.1 Simplistic Scenarios

The main aim of the simplistic scenarios was to assess the underlying mathematical model and its overall ability of identification of suitable situations. Results and characteristics of the three simulations performed are listed in Table 2.



**Table 2.** Results of simplistic scenario experiments.



Simulation Result	Time	Description
	30 days	<ul style="list-style-type: none"> <li>- One situation class (time of day)</li> <li>- User prefers to read news every morning</li> </ul>
	30 days	<ul style="list-style-type: none"> <li>- Two situation classes (time of day and weather)</li> <li>- Weather contained two different situations</li> <li>- User prefers to read news every morning</li> </ul>
	30 days	<ul style="list-style-type: none"> <li>- Two situation classes (time of day, day of week)</li> <li>- Date of week class contained seven different situations</li> <li>- User prefers to read news every morning</li> </ul>


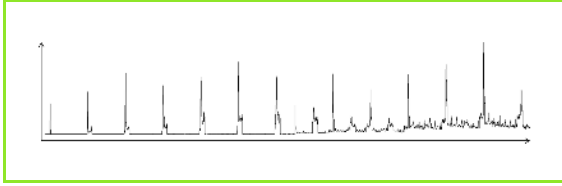
The results in simplistic scenarios show the correctness of our method's mathematical model in these conditions. It is apparent that with addition of unimportant situation class is our method adjusting a slightly longer which is to more extent between second and third simulation. In all the three cases the local maxima that represent most suitable situations identified by our method are easily recognized and correspond to the mornings for the timespan of the simulations.

### 6.1.2 Complex Scenarios

Main aim with the complex scenarios was to verify the abilities of our method to work and correctly identify situations suitable for recommending actions in more complex simulated environments engaging various types of users. The results and characteristics of the four simulations for complex scenarios are listed in Table 3.

**Table 3.** Complex scenario simulation results.

Simulation Result	Time	Description
	30 days	<ul style="list-style-type: none"> <li>- Nine situation classes</li> <li>- User prefers to read news every morning</li> </ul>
	90 days	<ul style="list-style-type: none"> <li>- Nine situation classes</li> <li>- User prefers to read news every Monday morning</li> </ul>

	90 days	<ul style="list-style-type: none"> <li>- Nine situation classes</li> <li>- User prefers to read news every Monday morning and Friday evening</li> </ul>
	90 days	<ul style="list-style-type: none"> <li>- Nine situation classes</li> <li>- User prefers to read news every Monday morning with change to Friday evening</li> </ul>

The simulations listed in this section build on top of each other and the results show that our method is capable of handling even cases when there is up to eight unimportant situation classes and only one important (first simulation), in scenario where the scenarios are important only when present together (second simulation), in scenario where more important tuples are (third simulation) and also in case of a preference change.

## 6.2 Live Experiment

During the live experiment six users about whom we were collecting context information provided us with feedback indicating whether or not is it a good time to read some news or do something similar. The feedback could be sent in four different notions:

- Certainly a good time
- Quite a good time
- Quite a bad time
- Very bad time

During the course of two weeks we have collected from these six users more than 700,000 situation observations and more than 3,000 unique feedback responses that unveiled two important characteristics of observation in the real world use.

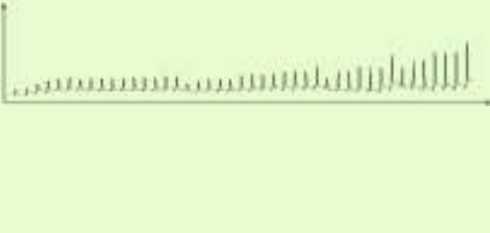

The first one is a kind of situation “lock-in” that occurs within situation classes where the situation does not change for a longer period of time. Such situation is for example temperature that is often fixed for a certain timespan with little or no change. In this case rules are created with one (locked-in) situation what in our case, when we do not know the background information of these situation classes, causes temporary drop in recommendation certainty after the situation is no more locked-in. These are however returned to their former level after a short period of time when our method correctly classifies unimportant situations.

The second important real-world characteristic is random indications that occur sparingly and do not represent any situation – action link considering the situation classes being observed and available. It is possible that this feedback depends on a part of user’s context with which our method does not work because the client service does not provide such context information. In both cases it is important that we are able to identify the most significant suitable situations even with significant noise in the data.

### 6.3 Additional Simulations

The aim of the additional simulations was to evaluate on how our method can handle the two specific characteristics identified within the real-world use. The results of both simulations and their characteristics are described in Table 4.

**Table 4.** Additional simulation results.

Simulation Result	Time	Description
	42 days	<ul style="list-style-type: none"> <li>- Only one temperature situation for the first two weeks</li> <li>- Two additional situations added in weeks three and four</li> <li>- Another additional situation added for the last two weeks with the first one not being used anymore</li> </ul>
	90 days	<ul style="list-style-type: none"> <li>- 7 to 14 random indications (feedbacks) in every week</li> </ul>

The results of additional simulations show that our method is capable of working correctly even in these cases with difficult scenario characteristics as the situation lock-in and random indications both are. In the case of situation lock-in (first simulation) a drop in recommendation certainty can be spotted at the beginning of third and fifth week just as we expected, however peaks are still distinguished clearly enough and thus suitable situations can be identified. In case of the random feedbacks despite the relevant feedback constituting for only almost one third of all the feedback, thanks to our aggregative rule model we are still able to reliably identify the suitable situations.

## 7 Conclusions and Future Work

We live in a world of pervasive computing where automated technology begins to be a part of our everyday lives. Most of the autonomous computing today however features one-size-fits-all principles where they do not distinguish the differences between individuals. A similar problem is in the domain of proactive news recommendation or interesting content provision in general where methods usually only consider the content and not for example if it is the right time to disturb the user with some or a particular type of content.

We have designed and in this paper described our method for aiding with the autonomy of the pervasive services and applications by enhancing them with action recommendations based on situation rules that our methods defines from the observations and feedback received.

Evaluation of our method shows that it is capable to operate in a wide range of scenarios, from those that only require a small portion of user's context to those where a considerable amount of noise is involved or where users change their long-term behavior.

Our future work mainly involves enhancing the capabilities of our method to handle unknown variables and hidden links. This involves reducing the noise, identifying steady situation classes and characteristics of any situation class in general from the observations to better approximate their importance and solving the sub-model within model issue. Other enhancements include a support for collaborative models, where rules can be transferred and shared among similar users just as content is recommended in this collaborative way. This may significantly counter the known cold-start problem for new users, after a sufficient user-base is already active in the target service.

We have designed our method in a way that it is easily portable to any domain and the only responsibility end-user services and applications have is deciding on what actions they will be performing and what context information can be relevant to support the decision process. We believe that our method and frameworks based on it will due to their simplicity encourage more services and applications to become context-aware and more intelligent with it as well.

## Bibliography

1. Simpson, J., Weiner, E., eds.: The Oxford English Dictionary 2nd edn. Oxford University Press, Oxford (1989)
2. Böhmer, M., Bauer, G.: Exploiting the Icon Arrangement on Mobile Devices as Information Source for Context-awareness. In : MobileHCI, Lisboa, pp.195-198 (2010)
3. Krüger, A., Baus, J., Heckmann, D., Kruppa, M., Wasinger, R.: Adaptive Mobile Guides. In : The Adaptive Web: Methods and Strategies of Web Personalization. Springer-Verlag, New York (2007)
4. Korpipää, P., Häkkinä, J., Kela, J., Ronkainen, S., Käsälä, I.: Utilising Context Ontology in Mobile Device Application Personalization. In : MUM, Maryland, pp.133-140 (2004)
5. Ala-Siuru, P., Tapani, R.: Understanding and recognizing usage situations using context data available in mobile phones. In : ubiPCMM, California (2006)
6. Roberts, M., Ducheneaut, N., Beloge, B., Partridge, K., Price, B., Bellotti, V., Walendowski, A., Rasmussen, P.: Scalable Architecture for Context-Aware Activity-Detecting Mobile Recommendation Systems. In : WoWMoM, California, pp.1-6 (2008)
7. Billsus, D., Pazzani, M.: Adaptive News Access. In : The Adaptive Web: Methods and Strategies of Web Personalization. Springer-Verlag, New York (2007)
8. Cantador, I., Castells, P.: Semantic Contextualisation in a News Recommender System. In : Workshop on Context-Aware Recommender Systems, New York (2009)
9. Liu, J., Dolan, P., Pedersen, E.: Personalized News Recommendation Based on Click Behavior. In : IUI, Hong Kong, pp.31-40 (2010)



# Appendix E:

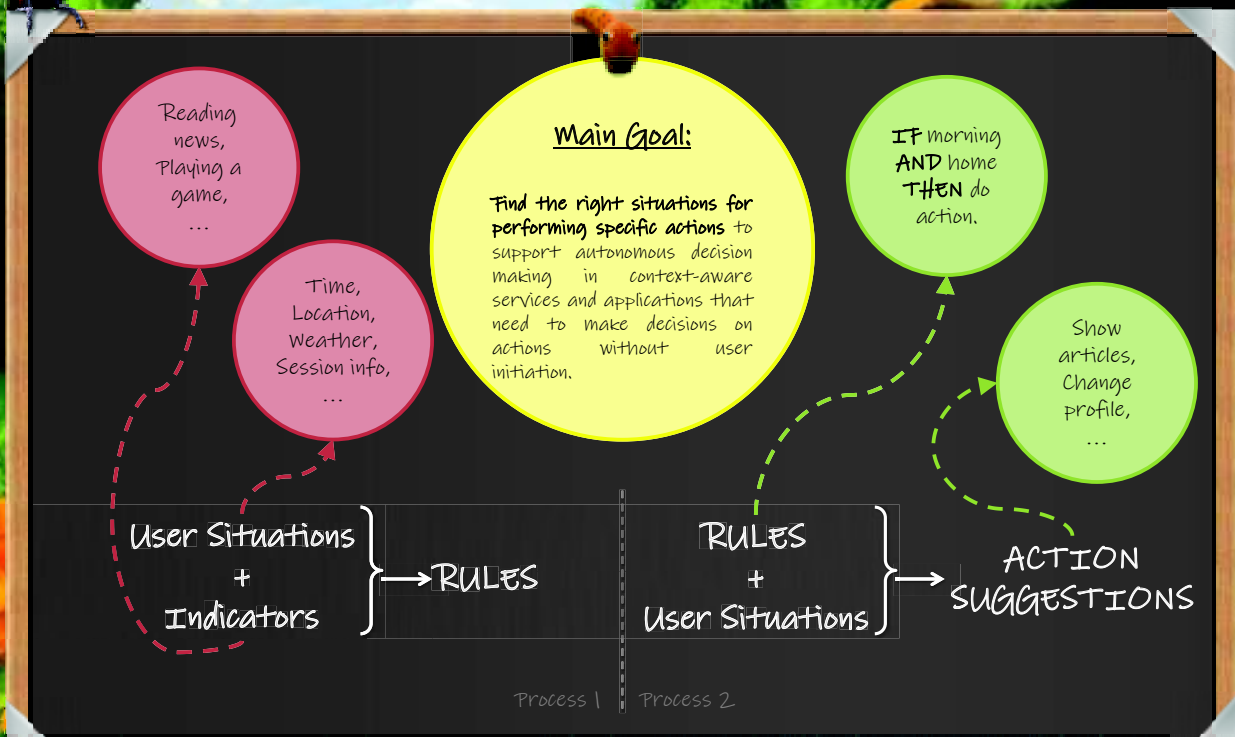
IIT.SRC 2012 Poster





# Action Recommendation Using Situation Rules

Anton Bencic & prof. Maria Bielikova



## Action Model

Set of rules  
Based on the user's performance (explicit and implicit feedback)  
Two parts

1. Antecedents (situation when the rule applies)
2. Conclusion (action that the rule suggests)

Representation through symbols  
Extended Time sensitivity principle

## Situation Model

Representation through symbols  
Certainty values  
Time sensitivity principle  
Main advantages:

- Domain independence
- Elimination of privacy issues
- Allowing collaborative models

## Evaluation

User reading news in the morning

Reading news on Monday mornings

Monday mornings and Friday evenings

Preference change

## Method for Action Suggestion

### 1. Rule Generation

Based on:

- Current user's situation
- Action indicator

Rules for all possible situation combinations  
Base certainty distributed accordingly

### 2. Recommendation Calculation

Using user's action and situation models  
Steps:

1. Computing the importance of situation classes
2. Modifying rule certainties
3. Computing the final recommendations





# Appendix F:

Contents of the Digital Medium



ToDo