Cascaded Generic XCS to Learn About Reminding Preferences

Nadine Richard
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku
Tokyo-to 101-8430, Japan
nadine@nii.ac.jp

Samuel Tardieu
ENST / ParisTech University
46, rue Barrault
75013 Paris, France
sam@enst.fr

Seiji Yamada
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku
Tokyo-to 101-8430, Japan
seiji@nii.ac.jp

ABSTRACT

We are developing an adaptive reminding system, which learns when and how to present notifications. In this paper, we focus on our XCS-based model, composed of two cascaded sets of classifiers: the first one learns a categorization of calendar data, while the second selects the appropriate forms of combinable reminders depending on the user and device contexts. After describing the characteristics of the input data, we present the extensions we propose to provide a generic XCS architecture, which seems suitable for processing those specific inputs. Finally, we describe our user feedback mechanism, and the according reward system.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*

General Terms

Algorithms

Keywords

Learning Classifier Systems, personal time management, adaptive reminders

1. INTRODUCTION

We are developing TamaCoach, an adaptive, emotional and expressive assistant for reminding tasks and events. This interface agent adapts to the organizational skills and preferences of a user, by learning when and how to present notifications, instead of requiring the user to set explicit alarms.

A reminder can be presented in different but combinable forms: a new item in the GUI list of pending tasks or events, a pop-up dialogue box, an e-mail message, a mobile e-mail message, or a sound alarm. The triggering of a reminder depends on:

- The relative temporal distance to the event starting date or to the task due date.
- Various attributes that help to classify user habits and preferences, *e.g.* contact information, priority or categories.

The form of a reminder mainly depends on the user status (availability, on-going activity, physical location, and mood) and on the capabilities of the host device. The necessary data about tasks and events are extracted from iCalendar files, which are produced by an external calendar or todo management application. We gather information about the current context of the user through our GUI, and information about the agent execution context through the operating system.

As users may interact infrequently with their calendaring application, designing an appropriate feedback system for the learning module is a crucial issue. The user interacts both explicitly and implicitly with TamaCoach, respectively through the dedicated user interface (GUI and e-mails), or by updating iCalendar data through his/her favourite calendaring application. In particular, the user can explicitly respond to a notification, in order give feedback about the usefulness of the reminding system.

Our XCS-based model is composed of two cascaded sets of classifiers: the first set categorizes tasks and events in terms of temporal distances, priorities, *etc.*, in order to decide whether a reminder should be triggered, while the second set evaluates the user and the device contexts, in order to select the suitable kinds of notifications.

2. RELATED WORK

2.1 Personal Time Managers

Adaptive systems like PTIME (as part of the PExA assistant) learn about user scheduling habits and preferences, mostly in order to autonomously negotiate meetings [1]. Like most of the investigated office- or healthcare-related personal assistants, PExA is a cognitive agent that learns and reasons about tasks, user behaviour and its own behaviour, in order to justify its actions, answer questions and give advice.

Autominder is an adaptive reminding system, intended for cognitively impaired elders, who prefer to live at home [2]. It learns about routine activities, and monitors their daily performance in order to issue reminders whenever an

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

essential task is not executed on time. AUTOMINDER is based on dynamic constraint solving, and focuses on detecting the discrepancies between a predefined plan and the actual performance of tasks.

2.2 XCS for Context Detection

Learning Classifier Systems have been extensively used for data-mining and for modeling animats. They have less often been applied to interface agents, especially to learn about user's preferences and habits. Recently, Shankar *et al.* have been experimenting the use of a classical ternary XCS to learn about the user context [5]. The authors first gathered data about when a subject preferred to be interrupted by a reminding system when performing a task. After a step of off-line mining of those data, their SYCOPHANT system was able to predict which reminder[1] would be suitable for an appointment, given the current user context (keyboard/mouse activity, state of main processes) and user's vicinity context (motion and speech detection).

2.3 TamaCoach

The objective of TAMACOACH is not to help users in maintaining a todo-list, but to learn when and how to remind them about what they have explicitly planned to do. However, such an application is a complement to more complex assistants like PEXA. As suggested by its name, the ultimate purpose of the TAMACOACH project is to investigate virtual coaching in the case of personal time management. Our mid-term goal is however to experiment on its adaptiveness and expressiveness. More information about the emotional and expressive aspects of our work can be found in [3].

Even if SYCOPHANT triggers adaptive reminders for appointments, this system focuses on the automatic detection of the level of interruptibility of the user, considering the internal and external contexts. Before selecting a set of reminders depending on the user context, TAMACOACH performs a categorization of current events and tasks.

3. DESCRIBING SITUATIONS

In order to trigger an appropriate reminder, it is necessary to extract relative temporal distances and additional values from ICALENDAR data, and to incorporate the context of both the user and the application. Most of the conditional attributes processed by our learning module correspond to fields provided by the ICALENDAR format. ICALENDAR data are extracted to be stored into a local database. This database also contains additional attribute values, computed from past experiences, *e.g.* the average delay in achieving a given category of task. Most of the values stored in the database take part in the decision process; in this case, before being sent to the Situation Manager, the values are discretized.

3.1 Calendar Data

ICALENDAR files mainly contain the following information about each task or event: absolute dates (start date, event end date, task due date), duration, recurrence rules, user-defined categories, alarms, progress status of a task, involved contacts, and various other characteristics (summary, priority, transparency, location, *etc.*). Some ICALENDAR fields, like the summary or the attached documents, are not significant enough for the learning process; their values are stored in the database only to be presented to the user within the reminders.

Categorical information like priority, transparency, user-defined categories, or involved contacts are directly extracted from the raw ICALENDAR data. Other values are computed from ICALENDAR] data before being stored in the database. In particular, the *relative temporal distance* is necessary to decide when to trigger a reminder: it corresponds to the duration between the current date, and the starting date of an event or the due date of a task. Relative temporal distances are computed from ICALENDAR dates, durations and reccurrence rules. The reccurrence rules are translated into separate occurrences, with absolute dates and durations.

To be relevant for the learning mechanism of our agent, the raw distance is approximated and discretized into a pair of symbolic values, the granularity and the distance, in order to express how close the current date is to the deadline or starting date. With such a decomposition, a calendar item is easily categorized as happening soon or in a long time. Keeping track of the granularity is necessary to consider relative symbolic values like *very close* or *far* in an appropriate time context.

For clarity and efficiency purposes, we chose a set of significant thresholds in order to categorize both the granularity and the approximate value of a distance, instead of proposing functions for computing this categorization. The granularity of an item can be *very short-term* (<24 hours), *short-term* (<10 days), *mid-term* (<42 days) or *long-term* (≥42 days). For each granularity, the temporal distance can take six values, from *very close* to *very far*, or can be assigned to *late*[2]. For instance, a *close* event will start within the next hour for a *very short-term* granularity, or within the next two weeks on a *mid-term* scale.

From the number and the priority of current tasks, extracted from ICALENDAR data, we also compute the current *task load* of the user. This value completes the information about the busy state of the user, which might not be accurate enough.

3.2 Historical Data

For learning a generalized classification of calendar items, we assumed that the most useful data are the user-defined categories assigned to an event or a task. Therefore, in order to learn about the user habits concerning a kind of activity, we propose to use additional attributes related to the categories of past events and tasks: the average duration and the average delay observed for a given category, plus a flag indicating whether the user is generally early when achieving this kind of task or attending this kind of event[3].

On the other hand, notifications can be repeated only if the user explicitly asks to be reminded later. As a frequent repetition of the same reminder will probably annoy the user, information about previous reminders enables the system to learn when to present again the same notification

[1]among four possible: pop-up window, voice reminder, both, or none

[2]This additional distance value can be used for each granularity, in order to express the severity of the delay since the starting date or the due date occurred.

[3]If this is the case, the category delay represents how long before the deadline this kind of task is usually achieved.

content. We thus also store the number of reminders already triggered for a given item, and the temporal distance since the last reminder.

3.3 User Context

The user status is the agregation of the following information: the *busy state* (available, busy, very busy, away, signed-off), the *mood* (very good, good, average, bad, very bad), the *activity* (work, vacation, commuting, sick, conference, *etc.*), and the *location* (office, home, transportation, *etc.*).

The domains of the location and the activity attributes are defined and extended by the user, while the busy state and the mood have a fixed set of values. Presently, the user status is gathered directly through the GUI. Nevertheless, it would be more reliable and less distractive to detect the user activity load automatically, through monitoring (keyboard/mouse activity, estimated posture, biosensors, *etc.*).

3.4 Device Context

The execution context of TamaCoach is composed of the *device* attribute (desktop, PDA, *etc.*), and two flags that indicate whether the host device has a network connection, and whether it can play sounds. Such information is required to adapt the selected actions to the execution environment: for instance, if the device has no network connection, the reminding system cannot send an e-mail.

4. CASCADED GENERIC XCS

The learning system of TamaCoach is composed of two dependent XCS modules, with different purposes. Both Classifier Systems are based on the same model of a generic XCS, extended to take into account the specificities of our input data. This model is implemented in Python, a portable, dynamic, efficient and pure object-oriented language. This language is particularly suitable for quick prototyping, and is available for the platforms we are experimenting on.

4.1 Two Cascaded XCS

TamaCoach learns about two distinct functions: when to trigger a reminder, and how to present this notification to the user. As shown in figure 1, the learning system is composed of two XCS-based modules, connected through the Situation Manager. The first Classifier System (CS1) categorizes the data concerning a given calendar item. Its output is a priority value, which indicates how urgently TamaCoach should trigger a reminder for the item; a priority of 0 means that no reminder should be presented. Then, if this priority is not 0, its value is used to create a situation vector for the second Classifier System (CS2).

Depending on the current context of both the user and the host device, and on the priority given by the first set of classifiers, CS2 is in charge of choosing the appropriate kinds of reminders to be produced. Because the possible reminder forms are compatible (*e.g.* TamaCoach can launch a pop-up window and send an e-mail for the same item), the output of CS2 is a set of actions to be performed. As CS2 needs CS1 output to process the given situation, CS1 and CS2 are said to be cascaded.

4.2 Any Discretized Value

The discrete values of conditional attributes and actions can be of any type, as long as the right method is available

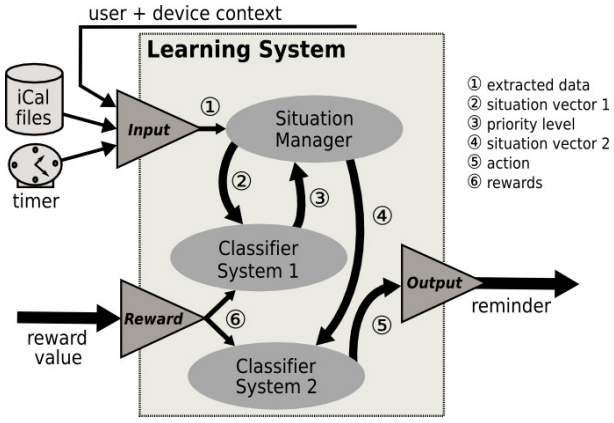

Figure 1: Architecture of the learning system.

for testing the equality between two objects. Attributes and actions can also take the Python `None` value, which corresponds to the # character used in classical ternary LCS. In our application, values are booleans, integers or character strings.

Each attribute is associated to a value domain. The form of the condition part of the rules is simply defined at the creation of the XCS by giving a list of the value domains. The set of action values is also defined when instanciating the system. Nevertheless, some domains can be dynamically modified, when referring to user-defined values: the calendar names, the item categories and locations, the contacts, the user locations and activities, the host device.

4.3 Situation Instances

Successive situations are produced by the Situation Manager. In case of multiple contacts and/or categories for a single item, a set of *situation instances* is created in order to let the Classifier System evaluate one percept for each contact or category. This solution is inspired from the work of G. Robert [4]: some attributes, *e.g.* the positions of surrounding enemies in a first-person shooting game, are considered like variables to be instanciated successively for the same situation.

A situation is thus a list of situation instances: this list is given to the XCS, which selects the most suitable action for each instance, and thus produces a list of actions matching the whole situation. If needed, this set of actions go through a compromise phase. In particular, for our application, such a compromise is necessary to compute a single priority value from the set of priorities chosen by CS1.

5. THE REWARD SYSTEM

5.1 Getting Feedback from the User

Together with a pop-up or an e-mail reminder, five possible replies are proposed to the user:

- *Accept*: the reminder was useful and the user will act immediately, *e.g.* by finishing the task or attending the meeting.
- *Later*: the reminder was useful but has to be presented again later, *e.g.* if the user cannot perform the task immediately.

- *Too early*: the reminder has to be presented again later, because it was issued too early.
- *Ignore*: the user does not care about that particular task or event.
- *Too late*: the user does not care about the reminder, because it arrived too late.

The items for which the user has replied *accept*, *ignore* or *too late* are considered as *acknowledged*, and will not be processed by the learning system anymore. A *later* or *too early* reply will trigger a new reminder later.

This light-weight, explicit feedback system enables TamaCoach to evaluate the quality of its actions, without disturbing busy users too much. Nevertheless, in order to keep the system useable, we cannot constrain the user to reply explicitly to each reminder: updates of the iCalendar data using an external tool can also be considered as an implicit response to a reminder. Any user reply causes the according reminders to be destroyed, but only explicit replies will be taken into account for computing reward values.

The user may not reply immediately to a reminder for various reasons: she forgot to update the user status before leaving the office, did not check her mobile e-mails, or is too busy to reply by e-mail or through the GUI. It means that a new notification can be issued even when a previous reminder is awaiting for the user's action. A notification will stay in the list of pending reminders until the user answers back, either explicitly or implicitly.

5.2 Computing and Distributing Rewards

The value of the reward is computed from the explicit reply given by the user to a specific reminder and concerning a specific task or event. All the rules that triggered notifications for the same calendar item will be rewarded, but the replied reminder will lead to a stronger positive or negative value to be sent to the involved set of rules.

The *accept* reply indicates that the user found the notification useful: the reward is thus maximal. The *later* response means that the notification was useful, but should be repeated at some point: the reward is positive. When answering *too early*, the user expresses that the notification should have been issued later: the triggering rules are thus slightly punished. The *ignore* reply means that reminding this kind of items is useless: the rules will receive a negative reward. The rules will get the maximal punishment if the user indicates that the notification arrived *too late*.

Because of the delayed, asynchronous replies coming from the user, the set of rules that triggered a particular notification needs to be stored within each XCS, until the appropriate reward is received. Therefore, we need to keep track of a set of rules, using a set identifier. However, a delayed reply should not be considered as a negative feedback, as we cannot know why the user did not react immediately. We thus just consider that, in case of a delay, the next reminders will not be selected by an up-to-date set of rules.

The user can update the iCalendar data independently from TamaCoach: if there are awaiting notifications concerning the modified item, they are simply destroyed. No reward can be computed, as implicit actions do not give any clue on the usefulness of the notifications. Therefore, we have extended the XCS model with a mechanism to *forget* about a set of rules, without giving a reward.

6. CONCLUSION AND FUTURE WORK

We have presented our adaptive reminding system, in particular the main features and the architecture of the XCS-based learning module. We have defined the requirements in terms of contextual input data, to be extracted from iCalendar files, or to be obtained from the user and from the host device. We also have described how to get useful but light-weight feedback from the user, in order to compute appropriate reward values for the learning system.

Our prototype has been developped in Python, with a Qt GUI, and is running under three different platforms: a desktop PC (GNU/Linux), a laptop PC (MS-Windows), and a Zaurus PDA (GNU/Linux). The system is currently being evaluated, using a basic user simulator and stereotyped calendaring data.

In parallel, we are investigating how to simulate various user profiles for evaluating and tuning the learning system. Thanks to this simulator, we expect to be able to provide pre-trained sets of classifiers, which will adjust to the reminding preferences of individual users. Afterwards, we will extend the prototype with an animated character, and conduct experiments with human users in order to validate our hypothesis about the possible influence of an adaptive, emotional and expressive reminding system on different kinds of users.

7. ACKNOWLEDGEMENTS

This research project is founded by the Japan Society for the Promotion of Science (JSPS).

8. REFERENCES

[1] P. Berry, K. Conley, M. Gervasio, B. Peintner, T. Uribe, and N. Yorke-Smith. Deploying a Personalized Time Management Assistant. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, May 2006. Hakodate, Japan.

[2] M. E. Pollack, L. Brown, D. Colbry, C. E. McCarthy, C. Orosz, B. Peintner, S. Ramakrishnan, and I. Tsamardinos. Autominder: An Intelligent Cognitive Orthotic System for People with Memory Impairment. *Robotics and Autonomous Systems*, 44(3-4), Sept. 2003.

[3] N. Richard and S. Yamada. An Adaptive, Emotional, and Expressive Reminding System. In *Proceedings of the AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants*, Mar. 2007. Stanford, USA.

[4] G. Robert. *MHiCS, une architecture de sélection de l'action Motivationnelle et Hiérarchique à Systèmes de Classeurs pour Personnages Non Joueurs adaptatifs.* PhD thesis, Université Paris 6, May 2005.

[5] A. Shankar and S. J. Louis. Learning Classifier Systems for User Context Learning. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, Sept. 2005. Edinburgh, UK.