Machine Learning in the Support of Context-Aware Adaptation

Vivian Genaro Motti, Nesrine Mezhoudi, Jean Vanderdonckt
LILab – Louvain Interaction Laboratory - Université catholique de Louvain
Place des Doyens 1 – Louvain-la-Neuve 1348

vivian.motti@uclouvain.be

ABSTRACT
Adapting user interfaces according to the context of use aims at improving the usability levels of an application and enhancing the user experience, mainly by optimizing the users' interaction and reducing their errors. However, given the significant amount of information involved, adapting UIs often demands complex inferences. Because the context information is extensive, it is hard to prioritize it to decide the best adaptation techniques. Moreover, dealing with recurrent trade-offs, e.g. adaptability vs. performance, is not simple. To aid the adaptation decisions, machine learning algorithms can be applied to support reasoning, inferences and also to deal with complex or fuzzy information. Although ML can provide several benefits for CAA, there is no agreed framework that aids developers in applying it. Thus, aiming to fill such a gap, this paper defines potential scenarios of CAA where ML can be successfully applied, presenting their common requirements and main trade-offs.

Author Keywords
Context-aware Adaptation, Adaptivity, Adaptability.

ACM Classification Keywords
D.2.2 [**Software Engineering**]: User interfaces. H.1.2 [**Information Systems**]: Human factors. H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems. H5.2 [**Information interfaces and presentation**]: User Interfaces – *User-centered design*.

General Terms
Human Factors; Design.

INTRODUCTION
Nowadays users of many different profiles are interacting with several device types and from varied environments, resulting in significantly heterogeneous contexts of use in which the interactions to take place. As such, it is not scalable for developers to implement dedicated versions of UIs and applications that are suitable for and accommodate all constraints and characteristics of each specific context of use. Therefore, in the current computational landscape, of ubiquity, mobility, pervasiveness and context-awareness, adaptation becomes an inevitable solution. However, efficiently implementing adaptation taking several contexts into account to match them with appropriate techniques is a challenge.

In this sense, machine learning (ML) as a domain capable of supporting the solution of complex problems, is able to provide significant help [Alpaydin, 2004], [Bishop, 2006], [Barber, 2010]. Although machine learning is able to provide important benefits for CAA, so far there is no guidance that effectively supports developers in finding the best approaches to solve their recurrent problems in CAA.

This paper presents a roadmap that guides stakeholders in the application of machine learning algorithms in the domain of context-aware adaptation. Potential application scenarios are exemplified, supporting the development of similar applications. We present a set of potential approaches that support the development of adaptive and adaptable applications in an optimal fashion. This roadmap aims at guiding the development of CAA in all its phases, considering different application domains, context information and scenarios of use. We also identified the main trade-offs commonly encountered and commons requirements for applying ML for context-aware adaptation.

This paper is organized as follows: the Section 2 discusses related works; Section 3 presents the roadmap; Section 4 presents common requirements and the design decisions; Section 5 discusses the contributions, and Section 6 presents the final remarks and the future works.

RELATED WORKS
Since the early 90's ML has been applied to support different façades of CAA. Although these works are dedicated to explore distinct applications of machine learning for context-aware adaptation, they are scattered, each one focusing on a specific application of adaptation at a time without a unified view of their potential benefits.

CAA can benefit from ML potential especially during two distinct phases: the inferences about context information, and the CAA design decisions. Examples of ML application during the inference phase include: finding patterns in user interaction histories and clustering data, as users' profiles

[Jennings and Higuchi, 1993]. Examples of ML applied for the design decisions of the CAA process include: predicting the user behavior [Mitrovic, 2007], [Mitrovic, 2009], automating tasks, learning about the user preferences, and continuously evolving the adaptation engine itself. Other ML algorithms also have been successfully applied to support CAA, for example:

• **Bayesian Networks:** as directed acyclic graphs, composed by nodes and arcs representing respectively variables and their relationships, Bayesian networks have been applied to model users and to predict their needs. As such, the system observes the user events, and responds to them by triggering new events [Horvitz et al., 1998].

• **Clustering:** consists in associating information based on the similarities found in its properties. Information that shares the same characteristics is located together composing a group called cluster. E.g. contents in webpages can be clustered according to their users' access, as performed in MLTutor, distinguishing trends in user interaction [Smith and Blandford, 2002].

• **Decision Trees (DT):** are composed by set of rules strategically and hierarchically organized. DT aid the representation, the selection and the classification of data. In [Vanderdonckt, 1999] and in [Eisenstein and Puerta, 2000] decision trees were applied to support the decision and selection of the most appropriate widget to build a GUI.

• **Fuzzy Logic:** recommended to deal with fuzzy data, the fuzzy logic was applied to adapt services in a mobile context [Cao et al., 2005], and to evaluate design decisions of adaptive composition of mobile applications [Desruelle et al., 2011].

• **Genetic Algorithms:** Acay (2004) recommends genetic algorithms in adaptation for complex systems, since they are capable of handling highly constraint problems, they are scalable for higher dimensions, and also because in complex task the feature space can grow exponentially with the number of features.

• **Markov Models:** In a simple markov model each node lists possible states of the system, and the transitions reflect probabilities to change from one state to another. Only the last state (user action) is considered, however in second-order models the last two states (user actions) also count, [Deshpande and Karypis, 2004]. In CAA, markov models have been used to predict user actions based on their interaction history.

• **Neural Networks:** as an interconnected group of artificial neurons, by using processing elements to connect input nodes with output nodes, neural networks were applied to compile a user profile and guide web browsing according to the relevance of the web content [Seo and Zhang, 2000].

• **Rule Induction:** consists in extracting formal rules based on observation of data, or local patterns. It was specifically applied in MLTutor to suggest users relevant hypertext pages [Smith and Blandford, 2002]. Although it is the most common approach adopted for context-aware adaptation given its simplicity, it has also reduced expressiveness.

It is important to investigate how each machine learning algorithm support specific phases of context-aware adaptation in depth; but so far there is no unified view that supports the application of ML for CAA in a broad manner, covering general-purpose implementations. To contribute in this sense, we abstracted potential applications, defined common requirements, and highlighted the main trade-offs.

LEARNING GOALS

Based on the analysis of the related works presented above, we notice that the learning mechanism can be employed to achieve specific goals. These goals are achieved either based on patterns that are identified from the user interaction or based on the user evaluation of the adaptation results provide by the system. Generally these goals belong to 5 main classes:

Associating: for instance when a navigational pattern is identified in the user interaction history (recorded by means of log files), the tasks that are executed together can be associated in chunks, facilitating their access and making the interaction more efficient. In this way groups of tasks are created. One possible category is *favourite* corresponding to the most accessed contents or tasks. The same approach is also valid for groups of users, i.e. users with same interests and profiles can share their favourite contents.

Sorting: consists in abstracting the sequence of interaction of the user in order to re-define the access order of the tasks, for instance with a given user always access the menu items in a given order (this pattern must be also identified by analysing log files of the interaction history), this information must be used as the criterion to re-arrange the order of the menu items, links, buttons of the UI.

Suggesting/Recommending: based on expected interaction, the system can suggest to the user contents or tasks that are more likely to be accessed by them. A certain user profile can help to decide the contents of interest.

***Hiding/Deleting*:** the contents or tasks that have the least frequent access must also have less priority of access in the UI, i.e. while the access to the most common tasks and contents must be facilitated the least common tasks and contents can be grouped and accessed by means of an extra resource (link, button or icon). Thus, optimizing the space usage in the UI. Deleting the shortcuts to least accessed tasks can also be an option to optimize the space usage in the UI, by giving priorities to the most used contents and accessed tasks.

Creating: some interactions that are always repeated by the user while accessing and interacting with the application can be automated. For instance if the user always re-size (maximize) the window of the UI, the system can change the default settings to automatically maximize the window when the application is launched.

Aiming to fulfill the goals abovementioned, different ML can be employed. The next section illustrates how algorithms can be applied to perform CAA.

A ROADMAP OF MACHINE LEARNING APPLICATIONS

Although ML has already been applied to support CAA, the works so far are sparse and thus not integrated yet, given that there is no agreed framework capable of guiding stakeholders in the selection and in the implementation of ML algorithms to provide efficient CAA to the end users, in this section we provide a unified view of how ML algorithms can be employed for applications in the different phases of a CAA process.

Table 1. Guidance Roadmap defining ML algorithms according to the CAA goals

	Context:	**UI Generation:**
Decision Trees	- classify users according to their profiles (e.g. proficiency level, interests)	- widgets' selection - best modality decision
Clustering	- group users with similar characteristics (e.g., with profile clusters) - find clusters according to the device type (e.g. to tackle the mobile fragmentation problem)	- the menu items can be composed according to the frequency of access - the contents can be clustered regarding the profiles of the users interested (recommendation / suggestions)
Markov Chain	- the context changes can serve as a trigger for the adaptation techniques, i.e. given a change in the context one technique must be applied; e.g. when orientation is changed, the UI is rotated	- establishes a transition matrix with probabilities that can be applied for: each adaptation technique (and consequently each adapted UI) given a specific context; or each task, content within a UI
Regression	- identify the existence of patterns between user preferences (as common interaction histories) and adaptation techniques (e.g., change the application modes: silent, loud, etc.)	- association of widgets properties (e.g., dimensions, visibility, contrast levels) X evaluation metrics (e.g., ergonomics, aesthetics) per context of use

For each scenario listed in Table 1, specific refinements are needed to concretely model and implement the solutions. For instance, regarding the application of decision trees for UI generation, in the definition of the best modality, attributes gathered from the context information can be considered, as: occurrence of visual impairments of the user, availability of input and output devices (e.g. microphone, speakers, headset), battery level (in case of mobile devices). As possible classes for the DT, two modality types are envisaged: graphic and audio. It is out of the scope of this paper to provide further implementation details about modeling each scenario suggested, however we do provide in the next section further refinements about the design decisions.

DESIGN DECISIONS

Potential trade-offs. Although ML algorithms provide a set of benefits for CAA, care must be taken to not disturb the user interaction. Clearly, there are several trade-offs that can be expected by automating UI changes by means of adaptation. Therefore if the user interaction is not very well understood, the adaptation results can: bother, annoy, confuse or even prevent users from achieving their actual goals. That is why users must be always able to evaluate the adaptation, to confirm that the results achieved are convenient according to their goals and interests. Moreover, by changing the UI layout that the users are familiarized with, can make them lost, thus all the changes must be clearly indicated. The common requirements presented below aim at fulfilling CAA goals with ML, but also avoiding potential trade-offs.

Common Requirements. Technically, the learning process by means of ML algorithms occurs when: (i) the priorities (or weights) of the adaptation rules are modified in a decision tree (the conditions in each node), providing adaptation results that are more suitable for the context of the user, (ii) the records of the previous interaction behavior are analyzed and patterns are identified helping the Adaptation Engine to automatically perform new adaptations, (iii) a negative feedback is given by the user, and then an adaptation technique has its priority or weight decreased in the adaptation engine.

All these approaches rely on the end user behavior and feedback. For instance, if a certain pattern is identified in the log of the user interaction, it is likely that the navigation in the application can be optimized accordingly. And if the user accepts the adaptation proposed by the system, this feedback is used by the engine to set the adaptation as successful.

Machine learning algorithms aim at optimizing the adaptation process. Then, mainly the learning infrastructure involves two general requirements. *First* gathering the preferences of the user (with feedback or history of interaction for instance) and possibly other context elements and *then* adapting the engine (for instance by adjusting parameters related with priorities or preferences). These requirements are cyclic, i.e. once the adaptation is performed and present, the user intervene again providing his or her feedback, which, if positive, concludes the cycle, and if negative, adjust the engine. Such general requirements can be refined in four specific requirements:

- First, the user interaction needs to be recorded (with log files, or history features).

- Then, the adaptation process may request the authorization of the user to proceed, by announcing what will be done, and proceeding in case of agreement.
- The adaptation must be presented in a progressive fashion, and clearly stating to the end user that there is a change, and explaining *what* is happening, and also *why* it is happening.
- And finally, after the adaptation was performed, the user must have the option to:
    - Accept or reject it
    - Undo it
    - Evaluate or classify it, providing a feedback that will be used to adjust the parameters of the techniques implemented.

The user feedback must be applied to improve and evolve the adaptation engine, making its results more precise and accurate for next applications.

CONTRIBUTIONS

The main contributions of this paper include: motivating the use of ML algorithms for CAA; contextualizing application scenarios; and providing a roadmap to guide stakeholders in their design decisions regarding CAA supported by ML. The specific contributions include: the literature review of current related works; the description of ML algorithms applied for CAA scenarios; and the presentation of main trade-offs and common requirements in this domain.

FINAL REMARKS

Due to the large availability of technological devices, it is each day more important to implement and provide users applications able to effectively adapt themselves according to the context. Such implementation is a complex task, since much context information is involved and several application resources can be subject to context-aware adaptation. Therefore, ML algorithms can efficiently support this task. ML has as main advantages the ability to sense the context in a dynamic manner to evolve the adaptation engines, providing users more suitable adaptations. This paper presents a roadmap that can effectively guide stakeholders in the application of ML algorithms for CAA.

ACKNOWLEDGMENTS

This work received funding from the European Commission's Seventh Framework Program under grant agreement number 258030 (FP7-ICT-2009-5).

REFERENCES

1. Acay, L. (2004) Adaptive User Interfaces in Complex Supervisory Tasks. PhD Thesis.

2. Alpaydin, E. Introduction to Machine Learning, The MIT Press, October 2004, ISBN 0-262-01211-1

3. Barber, D. Bayesian Reasoning and Machine Learning, Cambridge University Press , 2010

4. Bishop, C. M. Pattern Recognition and Machine Learning, Springer, 2006, ISBN: 978-0-387-31073-2

5. Cao, J.; Xing N.; Chan, A.T.S.; Feng Y.; Jin B.; "Service adaptation using fuzzy theory in context-aware mobile computing middleware," Embedded and Real-Time Computing Systems and Applications, 2005. Proceedings. 11th IEEE International Conference on , vol., no., pp. 496- 501, 17-19 Aug. 2005 doi: 10.1109/RTCSA.2005.93

6. Deshpande, M. and Karypis, G. 2004. Selective Markov models for predicting Web page accesses. ACM Trans. Internet Technol. 4, 2 (May 2004), 163-184. DOI=http://doi.acm.org/10.1145/990301.990304

7. Desruelle, H., Blomme, D., and Gielen, F.. Adaptive Mobile Web Applications: A Quantitative Evaluation Approach. In Proceedings of ICWE. 2011, 375-378.

8. Eisenstein, J. and Puerta, A.. 2000. Adaptation in automated user-interface design. In Proceedings of the 5th international conference on Intelligent user interfaces (IUI '00). ACM, New York, NY, USA, 74-81. DOI=10.1145/325737.325787 http://doi.acm.org/10.1145/325737.325787

9. Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. (1998). The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. Proceedings of the Fourteenth Conference on Uncertainty in AI

10. Jennings, A. and Higuchi, H. (1993). A User Model Neural Network for a Personal News Service. User Modeling and User-Adapted Interaction 3, S. 1–25.

11. Mitrovic, N., Royo, J. A. and Mena, E. (2007). Performance analysis of an adaptive user interface system based on mobile agents. In Proc. DSV-IS '07. New York: ACM Press, 29--44.

12. Mitrovic, N. Royo, J. A. and Mena, E. (2009) Adaptive Graphical User Interfaces: An Approach Based on Mobile Agents, Networks, pp. 1-36, 2009.

13. Seo, Y.-W., & Zhang, B.-T. (2000). Learning user's preferences by analyzing web-browsing behaviors. In Proc. of int'l conf. on autonomous agents 2000 (aa '2000) (pp. 381– 387).

14. Smith, A. S. G. and Blandford, A. N. N. (2002) MLTutor : An Application of Machine Learning Algorithms for an Adaptive Web-based Information System, International Journal of Artificial Intelligence in Education, pp. 1-22, 2002.

15. Vanderdonckt, J. (1999). Advice-giving systems for selecting interaction objects. Proceedings User Interfaces to Data Intensive Systems, 152-157. IEEE. At: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=79147