Describing User Actions in Adaptive Interfaces with Probabilistic Acceptors

Matthias Bezold
Institute for Information Technology, University of Ulm
and Elektrobit Automotive Software, Erlangen
matthias.bezold@uni-ulm.de

Abstract

An important aspect of adaptive systems is the description of the user-system interaction, which can be used to derive new information about the user and to trigger adaptations, for instance by means of adaptation rules. In this paper, we present an approach that describes user actions by means of probabilistic deterministic finite-state automatons (PDFA), which are generated from an annotated corpus of user interactions. Based on a training set, different acceptors are created from recording data and can be employed by an adaptation framework to trigger adaptation rules. An evaluation of this approach with a prototype of an interactive TV system is presented.

1 Introduction

When adapting a system to user behavior, the description of the user-system interaction plays a crucial role. The system observes the user behavior and updates the user model or triggers adaptations of the system based on these observations. Therefore, an accurate and comprehensive means of describing user actions is required for adaptive interfaces.

The user-system interaction can be seen as a sequence of low-level events, such as button presses or speech utterances. A log component records these events and either stores them to a log file or processes them in real-time. For example, if the user presses a red button on a remote control, a certain action is triggered, e.g. opening a result screen in a digital TV system. From the low-level log data, it is not directly comprehensible which action was performed, i.e., a red button press can be seen, but not that it opens the result screen. A speech utterance can open the same menu, creating a completely different log entry for the same action. Therefore, a more intuitive and tangible high-level description for user actions is needed.

One approach of describing user behavior based on low-level system events are hand-crafted state automatons. Creating such automatons is a tedious and error-prone task and has to be performed by a person who has detailed knowledge about the system, such as the system developer. Statistical user modeling is one approach of describing the user. In [Winkelholz and Schlick, 2004], variable length Markov chains are used for finding regularities in user traces. [Levin *et al.*, 2000] use a Markov decision process for modeling the user-system interaction as a basis for learning and adapting the dialog strategy in a spoken language system. Our approach does not use the statistical description of the user interaction for adapting the system directly, but to trigger adaptation rules instead. Rather than predicting user actions, this approach provides a method for extracting meaningful interaction sequences from the user-system interaction in multimodal systems.

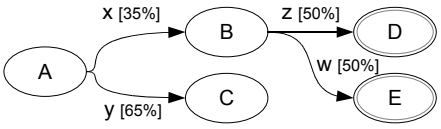

Figure 1: An example of a probabilistic deterministic finite-state automaton.

In this work, an approach is presented that describes user actions using probabilistic state acceptors, which are trained from recording sessions where sequences of log events have been annotated. These annotations are used to train the acceptors, which are then used either with log data or in a running system. Finally, the acceptors trigger rules in an adaptation framework.

This work is carried out in the context of multimodal systems, i.e., systems that have a graphical interface and can be controlled by a haptic input device and speech input at the same time. Recordings created for the evaluation of a prototype of a speech-enabled system, developed in the DICIT project [Matassoni *et al.*, 2008], are employed to assess this approach.

This paper is outlined as follows. First, the background of this work is introduced. Next, the application of probabilistic automatons to user action classification is introduced and an evaluation is presented. After explaining how adaptation rules are triggered in an adaptation framework using this approach, future work is discussed.

2 Background

This section introduces probabilistic deterministic finite-state automatons and the data used for the evaluation of this work.

2.1 Probabilistic Deterministic Finite-state Acceptors

Probabilistic deterministic finite-state automatons (PDFA), described in detail in [Vidal *et al.*, 2005a], consist of states and transitions between these states. A state change is triggered by a certain event, which occurs with a certain probability. An example is given in Fig. 1. In state 'A', the event 'x' triggering a transition to state 'B' occurs with a probability of 35 %, whereas the event 'y' going to state 'C' occurs with a probability of 65 %. Every state can be a final state with a certain probability.

```
[1181741700770] button {PowerOn}
```

Figure 2: Exemplary log line, showing a timestamp and a button press event (class "button") of the "PowerOn" button.

A state acceptor is a special case of the state automaton that is used to check if an automaton matches a given sequence. Starting at the initial state, the respective transition for every element of the sequence is taken. If none is found, the automaton does not accept the sequence. Otherwise, this step is repeated until a final state is found. The probability of the accepted sequence for this automaton can be computed by multiplying the transition probabilities of all used transitions. Therefore, the acceptance probability of the automaton in Fig. 1 for the sequence (x, w) is 17.5 %.

A PDFA can be generated from a set of sequences as follows (cf. [Vidal *et al.*, 2005b]). Starting at an initial state, a transition and a new state are added to the automaton for each element of the sequence. If the transition already exists, its weight is increased instead. After the automaton is constructed, the probabilities of every transition can be computed from the transition weights.

2.2 The Log Data

The presented approach was tested using a set of log files from evaluation sessions of a prototype of the DICIT system [Matassoni *et al.*, 2008], which is a speech-enabled interactive TV system including an electronic program guide (EPG). During the user evaluation of the system, log data of user sessions was collected by a logging component of the test system.

Every log file is a sequence of system events, such as remote control, speech input, or system state changes. An event consists of a timestamp, a type, and a number of parameters that are different for each type. An exemplary log line is given in Fig. 2.

3 Describing User Behavior using PDFAs

User actions are represented by certain log data sequences, e.g. pressing the "up" and "down" buttons for scrolling. Therefore, describing user behavior using PDFAs comprises the following steps. First, recordings need to be performed in order to create log data. This log data can then be annotated by assigning so-called interaction classes to certain sections of the recording. A PDFA can be created from these annotations for every interaction class. Optionally, the PDFAs can now be assessed with additional log data to evaluate them by annotating the additional log files using the PDFAs and comparing them to manual annotations. Finally, the PDFAs can be employed by an interactive system to determine what the user is currently doing, which requires a component in the interactive system that can apply the PDFAs to live log data.

3.1 Annotating the Log Data and Creating the PDFAs

In order to create the PDFAs for different interaction classes, annotations have to be created first. The annotation was performed with a custom graphical evaluation and annotation tool [Wesseling *et al.*, 2008], which displays log events from a recorded session in timeline views. Sequences of events are selected in these timelines and an interaction class is assigned to every meaningful sequence. However, there are many different sequences that describe the same interaction class. For instance, the "ChannelChange" action, which changes the current channel, can be executed using different commands: pressing the "channel up" and "channel down" or number buttons on the remote control or using speech interaction by saying the channel number or name. Each of these interactions produces different sequences of log events for the same interaction class. In these recordings, 34 different interaction classes were identified. The sequence length of these interaction classes is between 2 and 28, with the average length being 4.4.

	number	**percentage**
same	519	82 %
different - not recognized	106	17 %
different - value	9	1 %

Table 1: Evaluation results of 5 log files with a total number of 634 user actions from 34 different interaction classes.

The annotated sequences are extracted from the log data and PDFAs are created as described in Sect. 2.1, with every event corresponding to a transition. However, the context, i.e., the screen in which an action was performed, is relevant for some interaction classes. For instance, the action associated with some buttons is different in each screen, and therefore, the information which screen was active when the button was pressed has to be included in the event. On the other hand, the context should not be included for patterns that are the same in different areas of the system, such as scrolling. Hence, the context is omitted for these patterns. The interaction class definition therefore contains the information whether the context should be included by adding the currently active graphical screen to the event parameters.

3.2 Matching

The purpose of matching is to determine the interaction classes of a log sequence by finding out which subsequences are accepted by the PDFAs trained before. For recorded log data, sub-sequences are created for every log position, whereas the latest event sub-sequence is taken for live user sessions. A match factor is computed for all PDFAs for these sub-sequences, and the pattern is accepted if the factor is above a certain threshold. Since the probabilities of the branches of long and complex patterns can be low and since longer patterns should be recognized preferably, the probability is multiplied with a length factor to obtain the match factor. The element with the highest match factor is selected if more than match was found for a certain sequence.

3.3 Evaluation

A preliminary evaluation of this approach was performed by annotating 10 log files, with 5 being used to train the PDFA models and 5 being automatically processed. The results of the comparison of the automatic and the manual annotations are shown in Table. 1. Even with this limited data set, a match rate of 82 % could be achieved. Differences can occur for different reasons. First, an action can be present in the manual annotation, but not in the automatic one, or vice versa (*different - not recognized*), meaning that an action did not occur in the training data or is misrecognized. Using more training data can therefore reduce these kinds of errors. Second, a wrong class can be found (*different - value*), which occurs only 9 times in the

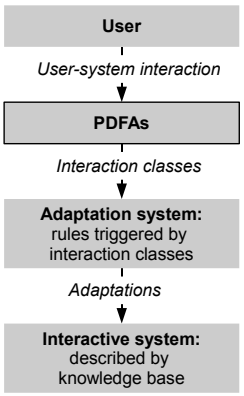

Figure 3: Overview of the system architecture.

checked recordings. These error can for instance occur if there are ambiguities in the annotations.

3.4 Limitations

The approach presented in this paper has limitations. First, as all statistical approaches, it highly depends on the annotations, i.e., only patterns occurring in the annotations can be recognized. For instance, if a certain action is triggered either by pressing the red button or saying "Record", but the red button is never pressed, then the red button is not included in the "Record" interaction class. In addition, the consistency of the annotations is very important. The more errors are in the annotations used for training the PDFAs, the lower the recognition rate will be.

Moreover, a list selection might consist of scrolling and pressing the OK button. In this case, the number and order of "up" and "down" key presses is relevant, which is not reflected well by the probabilistic nature of this approach. Therefore, the selection of every list item would need to create a separate event to make it applicable for this approach.

4 Using the PDFAs for Adaptations

Adaptive systems often employ rule languages to define the adaptations, for instance [Tran *et al.*, 2008] or [Dolog *et al.*, 2003]. The purpose of the PDFAs presented in this work is to trigger rules in an adaptation framework for interactive systems. The adaptation framework comprises a knowledge base that uses an ontology to describe the system (system model), the user (user model), and the user-system interaction (interaction model). By storing all information using the same formalism, it is available to the adaptations in a uniform way. The adaptations are defined as rules in a rule language that can exploit all the information provided by the knowledge base. Rule are used to update the knowledge base or to perform adaptations to the system.

In addition to the components of regular rules, which have a condition and a list of actions, the rules used in this framework have an event part that defines when the rule is triggered. One rule trigger type can an interaction class detected by a PDFA matcher, providing a comprehensive means to describe when the rules are executed based on the user-system interaction.

An overview of the adaptation architecture is given in Fig. 3. First, user actions generate event sequences, similar to the event sequences from log files. These sequences are then analyzed by the PDFA matchers and interaction classes are extracted. All adaptation rules that have the respective interaction classes as their triggers are evaluated and perform adaptations of the interactive system.

5 Conclusions and Future Work

An approach for describing the user-system interaction of interactive systems was presented, using interaction sequences as triggers for adaptation rules. First, log sessions are annotated to create a corpus of user interaction classes, from which PDFAs are generated. These matchers are then used in the interactive system to extract the known interaction patterns from the user-system interaction. An evaluation showed that even with a small data set of 5 training sessions and 5 test sessions, a match rate of 82 % could be achieved.

As future work, this approach could be extended to describe user behavior, which is more complex than describing user actions. However, behavior sometimes is characterized by the absence of events, e.g. if the user is idle because help is required. Therefore, the applicability of this approach to a more general kind of user behavior will be examined. Moreover, we will investigate how to use unsupervised learning instead of annotated log data.

A task model (cf. [Paternò *et al.*, 1997]) describes the possible user interaction on an abstract level. By combining it with the interaction patterns, it is possible to describe both what the user can do (task model) and what the user is actually doing (interaction model). For example, this enables the system to provide help if the user cannot finish a certain task or information about previously unused features.

In addition to adaptations, the PDFAs can be employed for the evaluation of interactive systems. More conclusions can be drawn from high-level information about the user-system interaction than from low-level events. For instance, task completion rates can be computed more easily if the task does not have to be described by low-level log data alone. Using an automatic annotation can reduce the evaluation effort.

References

[Dolog *et al.*, 2003] Peter Dolog, Nicola Henze, Wolfgang Nejdl, and Michael Sintek. Towards the Adaptive Semantic Web. In François Bry, Nicola Henze, and Jan Maluszynski, editors, *PPSWR 2003*, volume 2901 of *LNCS*, pages 51–68. Springer, Heidelberg, Germany, 2003.

[Levin *et al.*, 2000] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *Speech and Audio Processing 2000*, 8(1):11–23, Jan 2000.

[Matassoni *et al.*, 2008] Marco Matassoni, Maurizio Omologo, Roberto Manione, Timo Sowa, Rajesh Balchandran, Mark E. Epstein, and Ladislav Seredi. The DICIT project: an example of distant-talking based spoken dialogue interactive system. In *IIS 2008*, pages 527–533. Academic Publishing House EXIT, Warsaw, Poland, 2008.

[Paternò *et al.*, 1997] Fabio Paternò, Cristiano Mancini, and Silvia Meniconi. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In *INTERACT 1997*, pages 362–369. Chapman & Hall, London, UK, 1997.

[Tran *et al.*, 2008] Thanh Tran, Philipp Cimiano, and Anupriya Ankolekar. A Rule-Based Adaption Model for Ontology-Based Personalization. In Manolis Wallace, Marios C. Angelides, and Phivos Mylonas, editors, *Advances in Semantic Media Adaptation and Personalization*, volume 93 of *Studies in Computational Intelligence*, pages 117–135. Springer, Heidelberg, Germany, 2008.

[Vidal *et al.*, 2005a] Enrique Vidal, Franck Thollard, Colin de la Higuera, Franceso Casacuberta, and Rafael C. Carrasco. Probabilistic Finite-State Machines-Part I. In *Pattern Analysis and Machine Intelligence*, volume 27, pages 1013–1025. IEEE Computer Society, Washington, DC, USA, 2005.

[Vidal *et al.*, 2005b] Enrique Vidal, Franck Thollard, Colin de la Higuera, Franceso Casacuberta, and Rafael C. Carrasco. Probabilistic Finite-State Machines-Part II. In *Pattern Analysis and Machine Intelligence*, volume 27, pages 1013–1025. IEEE Computer Society, Washington, DC, USA, 2005.

[Wesseling *et al.*, 2008] Hugo Wesseling, Matthias Bezold, and Nicole Beringer. Automatic Evaluation Tool for Multimodal Dialogue Systems. In Elisabeth André, Laila Dybkjær, Wolfgang Minker, Heiko Neumann, Roberto Pieraccini, and Michael Weber, editors, *Perception in Multimodal Dialogue Systems*, volume 5078 of *LNCS*, pages 297–305. Springer, Heidelberg, Germany, 2008.

[Winkelholz and Schlick, 2004] Carsten Winkelholz and Christopher M. Schlick. Statistical variable length markov chains for the parameterization of stochastic user models from sparse data. In *SMC 2004*, pages 1770–1776. IEEE, 2004.