

From Behavioural Observations, to User Types, to Utility Functions

Bowen Hui & Craig Boutilier — Department of Computer Science, University of Toronto, Toronto, Canada

Abstract

This work describes the methodology in developing a model of a computer user based on behavioural observations. We adopt dynamic Bayesian networks to formulate the causal relationships between observed behaviour and user characteristics that are relevant to accepting automated assistance. The system infers the user's type and predicts the probability of accepting help. The system chooses the action with the maximum expected utility. To compute this value, we define a utility function that captures the variance in perceived rewards for automated help by different user types. In the future, we plan to learn the user's utility function.

Factors for Accepting Automated Help

We developed the model by identifying a typical user's values that are relevant to a user's response to automated help. The causal relationships in Figure 1 (left) shows that independence (I), quality of suggestion (QUAL), and current level of consideration (CONS) influences whether a person accepts help. Our concept of independence is one's preference to be autonomous when working. In turn, Figure 1 (right) shows that a person's current level of frustration (F), neediness (N), distraction (D), and independence influences the user's current level of considering the help. One's frustration level is an attitude towards automated suggestions, while one's neediness level is a task specific variable. Both of these values are **transient** and will change as the user progresses with the task and continues to interact with the system. On the other hand, distractibility and independence capture **static** traits of the user's behaviour while working in a computing environment.

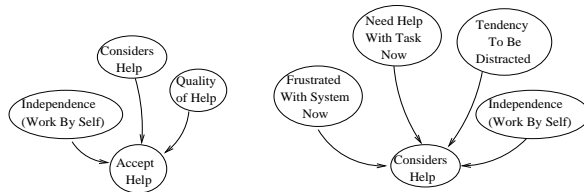


Figure 1: Variables for accepting automated help. Right: Causal relationships for accepting help. Left: Causal relationships for considering help.

Overview

We represent a user state as a potential function over the variables F, N, D, I. This potential is a distribution over the possible states of a user, which we denote as $BEL(F,N,D,I)$, and BEL for short. As the user interacts with the system, the system monitors fully observable behaviour from the user and use them to infer the user's current state and take appropriate actions. Since D and I are static values, they represent part of the user's **type**. The dynamics of the model will enable the system's beliefs about the user's type to converge to the true user type.

Assumption: User behaviour reflects user type.

Text Editor Suggestions: Word Prediction

In order to infer a user's state, we need to identify observations that are correlated with those states. Therefore, deriving the detailed structure of the model must be domain specific.

To model a wide range of user behaviour, we must consider different user skills and needs. We chose a text editor as a test-bed application because it is a common program for many computer users and its functions are also common to other communication software such as email and online chat. Furthermore, people with vocabulary and motor disadvantages often find that word processing software and communication aids allow them to concentrate on the quality of writing and give them a sense of authorship (Hasselbring & Glaser 2000). In the context of a text editor, the system would offer word prediction as its automated help. A sample interface is shown in Figure 2.

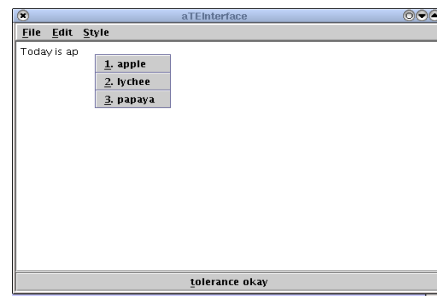


Figure 2: Mock-up of the interface for an adaptive text editor.

Deriving Fully Observable Variables

Assuming a typical computing environment, the source of fully observable variables come from keyboard and mouse events. Thus, we abstract these events into behaviour that would indicate positively or negatively with each of the user characteristics. The set of observations include:

- Frustration** – continuously pressing a key down, moving the mouse back and forth at a high rate, jamming into the keyboard, clicking the mouse button at a high rate, lowering the slider value
- Neediness** – erasing many characters via backspaces, browsing to look for help, pausing
- Distractibility** – browsing due to distraction, pausing
- Independence** – increasing the slider value, accepting help
- Browsing** – switching application windows at a high rate, surfing menus without selecting any menu item

In addition, we include responses to system suggestions, such as accepting help (acc), hovering over the suggestion box, pausing when the suggestion is available.

Dynamic Bayesian Model

By connecting all the relationships discussed above and including interface specific observations, we create a Bayesian network of adaptive help. Over time, user characteristics have temporal relations to their future counterparts, so we extend the model into a dynamic Bayesian network (DBN) as shown in Figure 3. DBNs (Dean & Kanazawa 1989) are founded on probability theory and Bayesian inference. Over time, variables with temporal dependencies are marginalized and rolled up into the next time step. Evidence is entered into the model and the interested potential is marginalized and normalized for its value. The system's policy is: *pop up a suggestion if $EU(POP) > EU(\neg POP)$.*

We set $EU(\neg POP) = 0$ as a reference point and compute $EU(POP) = EU(POP|OBS=acc)Pr(OBS=acc|BEL) + EU(POP|OBS=\neg acc)Pr(OBS=\neg acc|BEL)$.

Furthermore, $EU(POP|OBS=acc) = \sum R(F,N,D,I|Qual)BEL(F,N,D,I)$ and $EU(POP|OBS=\neg acc) = \sum C(F,N,D,I)BEL(F,N,D,I)$.

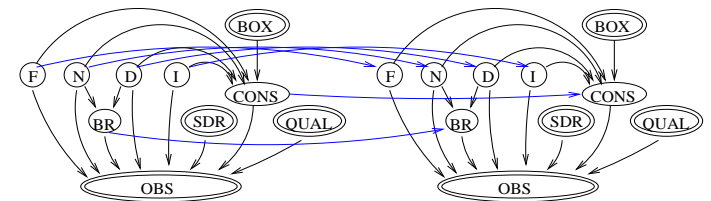


Figure 3: A two-time step dynamic Bayesian network model.

Simulation Results

In instantiating our simulation, we augment the DBN model in Figure 3 by adding two static user traits: TF – tendency to be frustrated with computers, and TN – tendency to need help. In our simulation, TF and TN are binary variables while D and I are tertiary variables. Together, we tested on 36 user types.

With pre-defined dynamics and model parameters, our simulation showed that the system's beliefs converged to the true user type in every case. The time it took the system to reach convergence varied from about 2 lines of text to 10 lines of text (on average 5 letters/word, 10 words/line).

A user's utility function can be defined in terms of the reward and cost function used in computing the system's policy. As expected, the simulation reports positive averaged reward for most of the 36 user types, with more needy and dependent users receiving higher overall utility and more frustrated, distractible, and independent users receiving lower utility. The next step is to learn the user's utility function.

References

- Dean & Kanazawa, 1989. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142-150.
- Hasselbring & Glaser, 2000. Use of Computer Technology to Help Students with Special Needs. *The Future of Children: Children and Computer Technology*, 10(2).

Acknowledgments

This project was in part supported by Precarn/IRIS, NSERC, and OGS.