

Computational Assistance – Learning When to Offer Help

Precarn/IRIS Student Scholarship Progress Report

Bowen Hui

Department of Computer Science, University of Toronto

`bowen@cs.utoronto.ca`

June 13, 2005

1 Introduction

In designing computational assistance, one must consider various domain variables as well as individual user differences that influence the effectiveness and transparency of assistance. In this paper, we document a causal analysis under the Bayesian framework for designing computational assistance. In particular, we adopt dynamic Bayesian networks to model the dynamics among the identified factors over time. To demonstrate the approach, we prototyped a word prediction application that interacts with the DBN user model. Unlike most word prediction software, the system does not propose a set of completion words every time the user types in a letter. Rather, it adapts to the style of the user and only makes suggestions when it believes that the user will benefit from the computed suggestions. Such a model is intended to model the styles of different groups of users and adapt to individual preferences. The methodology is generalizable to account for more complex software.

2 Design Issues

In this section, we discuss various design issues involved in this project. In particular, we separate the discussion related to the general methodology in Section 2.1 from the discussion related to domain-specific models in Section 2.2.

2.1 User Factors in Accepting Automated Help

We developed the model by identifying a typical user's attitudes and cognitive demands that are relevant to the response to automated help. User attitudes include one's frustration level with computers (F) and one's tendency to work autonomously on a computer task (I). User cognitive demands include one's neediness level (N) and one's tendency to be distracted while performing a computer task (D). In addition, the quality of automated help (QUAL) also contributes to one's likelihood to accept it. Altogether, the relationships are illustrated in Figure 1. Both the frustration and neediness 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 computing behaviour.

Thus, we propose to model the state of the user via the user's attitudes and cognitive demands at the task. To reflect the varying levels and combinations of these factors, we use 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 $BEL(F,N,D,I)$, or BEL for short. As the user interacts with the system, the system monitors fully observable behaviour from the user (cf. Section 2.2) and use them to infer the user's current state. Altogether, the relationships between the user variables and user behaviour make up the system's user model. Since D and I are static values, they constitute part of the user's type. The dynamics of the model will enable the system's beliefs about the user's type to approach to the true user type.

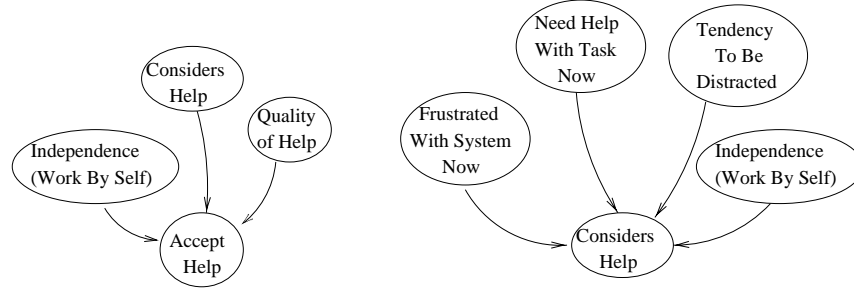


Figure 1: Variables for accepting automated help. Right: Causal relationships for accepting help. Left: Causal relationship for considering help.

2.2 Adaptive Word Prediction

In order to infer a user’s state, we propose 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 and word prediction software allow them to concentrate on the quality of writing and give them a sense of authorship [HG00]. In this setting, the computational assistance we are interested in is word prediction.

Assuming a typical computing environment, the source of fully observable variables comes from keyboard and mouse events. Thus, we abstract these events into behaviour that would correlate positively (e.g., jamming into the keyboard) or negatively (e.g., decreasing the adaptive slider’s value) with each of the user characteristics. The set of observations will expand according to the devices available in the system. For example, eye movement can be added if the system uses an eye tracking device. Browsing is aggregated as a separate variable because one cannot differentiate the underlying intention just by observing the behaviour. In addition, we include user’s responses to system suggestions, such as accepting help (acc) and hovering over the suggestion box.

3 Dynamic Bayesian Model

By connecting all the relationships discussed above and including interface specific observations, we create a Bayesian network (BN) of adaptive computational assistance. Over time, user characteristics have temporal relations to their future counterparts, so we extend the model into a two time-step, *dynamic* Bayesian network (DBN) as shown in Figure 2. DBNs [DK89] are founded on probability theory and BNs [Pea88]. Inference in a two time-step DBN is carried out in the same way as it is in BNs. 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.

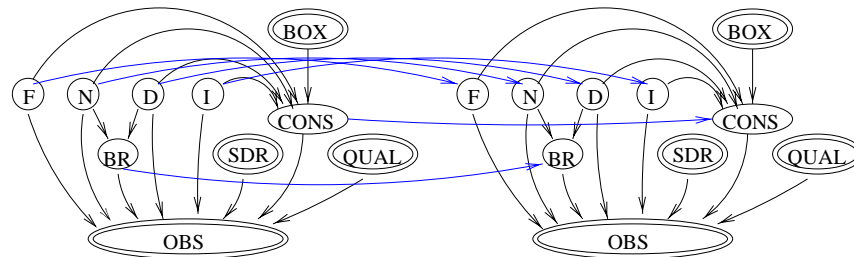


Figure 2: A two time-step DBN model. Observations are drawn in double-circles.

Figure 2 also includes variables from other system modules, which are BOX – whether a suggestion was made by the system, QUAL – the value of the assessed quality of the system’s suggestion, and SDR – the value of a slider widget for controlling the adaptivity that can be directly manipulated by the user. Under this design, the system’s task is focused on whether to offer assistance to the user while s/he is typing. The system has 2 actions – to offer a set of completion words, or to remain passive. Together, there are 729 user states (as defined by F, N, D, I, BR, CONS) and 1100 observation states (as defined by BOX, SDR, QUAL, OBS), yielding a total of 801,900 system states. The DBN model allows us to keep the representation compact in terms of the *factors* that are relevant in the domain, rather than using a *flat* state representation that is exponential in comparison.

3.1 Model Dynamics

In a DBN, each variable has an associated conditional probability distribution (CPD) that defines the dynamics from its parent variables. In particular, for every variable X, and its set of parents, Par(X), the CPD of X is $Pr(X|Par(X))$. In the current system, intuitive CPDs have been handcrafted for the purposes of demonstrating the overall concept. In Section 4.2, we discuss current experiments in learning these parameters from real users in controlled settings.

3.2 Belief State Monitoring

Since a CPD is defined over all parent values, CPDs are exponential in the number of parent variables in a discrete model. In a real time adaptive word prediction system, computations for monitoring the user’s belief state and propagating incoming evidence become an issue with large networks. The level of abstraction and representation also play a central role in computational efficiency. Although the belief state is small (a 3x3x3x3 potential function), temporal dependencies elsewhere in the network could result in a more complicated (clique) structure and thus, affecting algorithmic performance.

In our model, we have chosen an abstract representation that is intuitive from a designer’s perspective (e.g., model observations are behavioural patterns) and that reduces the number of temporal dependencies. Belief state monitoring is currently implemented in Matlab 6.5 R13. On average, this computation takes approximately 0.57 second on a Pentium M 1.2 GHz CPU 386 MB RAM. When abstraction is not feasible, approximation algorithms (e.g., [BK98]) can be considered.

3.3 Assessing the System’s Confidence

Standard word prediction software makes use of collocation statistics such as *n-gram probabilities* [SNN⁺01]. In particular, for $n = 2$, a *bigram* probability is $Pr(w_2|w_1)$, where w_1 is the previous word and w_2 is the current word being typed. Our system also adopts a bigram model, which is trained on 40% of the 100 million word British National Corpus (BNC). The system maintains the top 20,000 bigrams and 20,000 unigrams with backoff weights in its lexicon at runtime. We reserve 10% of the corpus for testing.

In assessing a completion word’s usefulness to the system, we attribute a utility to capture the number of characters saved from accepting the word. More specifically, we define the utility of a completion, c , with respect to the true word, s as $U(c|s) = \text{number of characters that are the same} - \text{number of characters to erase} - \text{number of characters to add to get to } s$.

Together, the utility and the probability measures are used to compute the *expected savings* of a set of completion words, $ES(c_i)$. For a suggestion with J completion words, we want each of the suggestions to be different from each other so that they each contribute a unique utility to the user. In other words, the overall set of suggestions should cover a larger area of probability mass. For K candidate words at runtime, the *joint* expected savings is, $JES(< c_1, \dots, c_J >) = [max(U(c_1|c_1), \dots, U(c_J|c_1)) \dots max(U(c_1|c_k), \dots, U(c_J|c_k))]$. P , where $P = [p(c_1|w_1) \dots p(c_k|w_1)]$. However, when $J \geq 2$, the number of comparisons increase exponentially. In the current prototype, we implemented a greedy algorithm to compute the JES at runtime that reduces computation to a linear number of comparisons in K . Finally, this quantity is used in the system as an observation QUAL that assess the quality of the language component.

3.4 Decision Policy

The system chooses the action with maximum expected utility. At any point in time, the system can pop up a suggestion (given the best completion words possible from the language module), or not. The system’s policy is: pop up a suggestion if $EU(POP) > EU(\neg POP)$. We define a reward function $R(F, N, D, I|QUAL)$ to indicate, for example, that users who are more independent appreciate the assistance less than those who are dependent by assigning lower values to independent types and higher values to dependent types. To model the opposite effect, we define a cost function $C(F, N, D, I)$ to associate, for example, a higher cost in interrupting users who are currently frustrated. These functions serve as *self-evaluation* metrics that is incorporated into the design of the system’s decision policy. Currently, the reward and cost functions are handcrafted, but we plan to learn them in the next phase of the project.

4 Experiments

To test the feasibility of the concepts underlying the model, we designed simulations documented in Section 4.1. Next, to replace the handcrafted parameters that describe the model dynamics, we are designing controlled experiments that explores different user states and logs corresponding user behaviour. Because user states are not directly accessible and cannot be explicitly elicited at every time step, we also discuss the use of a semi-supervised method for learning the data in Section 4.2. Finally, a usability study of the system will be briefly discussed.

4.1 Simulations

Simulations are necessary to test the effectiveness of this model because one cannot truly access values of a user’s state. In instantiating our simulation, we augment the DBN model from Figure 2 by adding two static user traits: TF - tendency to be frustrated with computers, and TN - tendency to need help. TF directly influences one’s current level of frustration. Similarly, TN directly influences one’s current level of needing help. In our simulation, TF and TN are binary variables while D and I are tertiary variables. Together, we tested on 36 user types. All of the experiments use an unseen portion of text from the BNC.

For each user type, we ran 100 simulations with 400 word long texts drawn from the test corpus. The averaged results show that the system’s beliefs converged to the true user type in all 36 cases. The time it took the system to reach convergence varied from about 20 words to 150 words (5 letters per word).

A system’s overall utility can be defined in terms of the rewards and costs it receives during the interaction with the user. We defined implicit reward and cost functions that vary according to the user’s current state (cf. Section 3.4). For each user type, we ran a separate simulation for 100 trials with 400 word long texts drawn from the test set. The general reward patterns reflect that the system adapts to the user’s responses – more acceptances encourages more suggestions, and vice versa. Across user types, the patterns also show that types that are more needy and dependent users receiving higher overall utility, while more frustrated, distractible, and independent users receive lower utility.

4.2 Learning Real User Behaviour

Many probabilistic systems use handcrafted parameters in modeling the dynamics of the system. While making use of domain expertise, these parameters are arbitrary and do not necessarily reflect real use. For systems involving human interaction, model parameters need to reflect the patterns of user behaviour. For these reasons, we designed an experiment to elicit such data from real users.

Because the task of typing is familiar to most potential participants, we designed a procedure that requires the user to type with a DVORAK keyboard. There were four conditions in the experiment. In the first condition, unnecessary delays of 2-5 seconds were randomly introduced into the system at fixed intervals. The second condition presented a mix of audio and visual distractors at regular intervals. These distractors have a lifetime of 7 seconds and can end earlier if the user closes their residing windows. In the third condition, the texts to be typed by the user contain a higher percentage of long words and esoteric vocabulary, as measured using the Fog index [Gun68]. In the last condition, a combination of the three settings were used. The purpose of this experiment is to log user behaviour with varying user states.

To assess the user's current state, questions to elicit the user's current F and N values were posed at the end of each sentence in all four trials. This online questionnaire yields a total of 16 combinations, which were presented in a random fashion. These questions allow us to learn the model's hidden variables in a semi-supervised fashion. A specific questionnaire with 18 items was designed to assess the user's general attitudes and tendencies while they are using computers. This questionnaire aims to elicit the user's type, as defined by TF, D, and I variables. The value of TN was indirectly measured via the user's typing speed (motor assessment) and the user's vocabulary rating of the text (cognitive assessment).

The learning task at hand is known structure with incomplete data. We plan to use the expectation maximization (EM) algorithm [DLR77] for approximating the CPDs in the model.

4.3 Usability Evaluation

A simple usability study will follow once the model parameters have been learned. The purpose of this study is to test the feasibility of the learned parameters in a realistic setting. A similar task from the previous experiment can be used to allow participants to become familiar with our adaptive system. Participants would compare our system's adaptive policy (as described in Section 3.4) with a system that always provides assistance and another that never provides assistance. As objective measures, the number of keystrokes made in each trial and the percentage of accepting system suggestions will be used. Participants will also be asked to fill out a questionnaire to report subjective responses.

5 Summary and Future Directions

We have developed a model of automated assistance and derived the various associating factors and overt behaviour in a computing environment. The model greedily computes a quality value of its assistance to the user. A self-evaluating utility function was defined in quantifying the amount of rewards and costs the system would receive during the interaction. A prototype has been implemented with initial simulations that demonstrate the concept.

The methodology described in this paper is general for modeling computational assistance. To truly assess the system's overall utility with respect to the user's changing state, the utility function needs to be learned with real users. In the current system, the utility function is a plausible distribution over different user states and system actions. The next step is to learn the structure of this function and update the distribution at runtime.

References

- [BK98] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, University of Wisconsin, Wisconsin, USA, July 1998.
- [DK89] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.
- [DLR77] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 1(39):1–38, 1977.
- [Gun68] Robert Gunning. *The techniques of clear writing*. New York: McGraw-Hill, 1968.
- [HG00] Hasselbring and Glaser. Use of computer technology to help students with special needs. *The Future of Children: Children and Computer Technology*, 2(10), 2000.
- [Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [SNN⁺01] Fraser Shein, Tom Nantais, Rose Nishiyama, Cynthia Tam, and Paul Marshall. Word cueing for persons with writing difficulties: Wordq. In *The Sixteenth Annual International Conference on Technology and Persons with Disabilities*, California State University at Northridge, Los Angeles, USA, March 2001.