A Personal Learning Apprentice

Lisa Dent[1], Jesus Boticario[2], John McDermott[3], Tom Mitchell, and David Zabowski

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
Lisa.Dent@cs.cmu.edu
Tom.Mitchell@cs.cmu.edu

Abstract

Personalized knowledge-based systems have not yet become widespread, despite their potential for valuable assistance in many daily tasks. This is due, in part, to the high cost of developing and maintaining customized knowledge bases. The construction of personal assistants as learning apprentices -- interactive assistants that learn continually from their users -- is one approach which could dramatically reduce the cost of knowledge-based advisors. We present one such personal learning apprentice, called CAP, which assists in managing a meeting calendar. CAP has been used since June 1991 by a secretary in our work place to manage a faculty member's meeting calendar, and is the first instance of a fielded learning apprentice in routine use. This paper describes the organization of CAP, its performance in initial field tests, and more general lessons learned from this effort about learning apprentice systems.

Keywords: learning apprentice, knowledge-based assistants, office automation

1. Introduction

Personal knowledge-based assistants have the potential to assist with many daily tasks. For example, a university faculty member might utilize various software assistants to manage courses, to organize the graduate admissions committee, and to manage a meeting calendar. Consider the example of a knowledge-based agent which provides an interface for editing a meeting calendar and advice such as when and where to schedule specific meetings, how long the meeting should last, whether to move an existing meeting to make room for the new meeting, whether to send electronic mail reminders to the participants, etc. The agent might also represent the faculty member in his or her absence by responding automatically to electronic mail requesting certain types of meetings. In order to be effective, such an agent would need considerable knowledge regarding the scheduling preferences and constraints of the individual faculty member (e.g., that meetings with graduate students are typically located in the faculty member's office, that meetings with undergraduates during office hours are typically allocated 30 minutes), as well as knowledge of various individuals within the environment (e.g., that Jon is an undergraduate majoring in computer science, that Raj is the Dean of computer science).

While such a knowledge-based assistant would be helpful, developing and maintaining such a system is difficult because (1) specifying the appropriate program action at design time is possible only for a very unambitious definition of "appropriate", (2) even if appropriate specifications could be obtained for one person, they must be redefined for each new user, and (3) even if one could specify the task and implement a program, maintaining the system in the face of an evolving world is costly. For these reasons, we believe that personalized knowledge-based software agents are not practical using standard software technology. Instead, we explore here the feasibility of learning apprentice systems, as defined in (Mitchell et al., 1985):

> We define *learning apprentice systems* as the class of *interactive* knowledge-based consultants that directly assimilate new knowledge by observing and analyzing the problem-solving steps contributed by their users through their *normal* use of the system.

More specifically, the thesis of the research reported here is that learning apprentice systems will enable practical development of a large class of personalized software assistants that are currently impractical due to high development and maintenance costs. The thesis rests on three key questions:

1. Can one construct interactive assistants that, even

---

[1]To appear in the *Proceedings of the 1992 National Conference on Artificial Intelligence*.

[2]Jesus Boticario is currently with the UNED, in Madrid, Spain.

[3]John McDermott is with Digital Equipment Corporation, Marlboro, MA.

when they are initially uninformed, are both sufficiently useful to entice users and able to capture useful training examples as a byproduct of their use?

2. Can one develop sufficiently powerful generalization methods to automatically acquire general decision-making strategies from this training data?
3. These generalization methods will require a stable set of attributes for describing training examples. Can this set be identified without incurring the same maintenance and customization costs we are attempting to avoid?

In this paper we support the above thesis by presenting and analyzing a particular learning apprentice for calendar management, called CAP. CAP has been in routine use since June 1991 by one secretary in our work environment. During this time it has captured hundreds of training examples and has learned to provide useful advice as the user adds new meetings to the calendar (e.g., suggesting the meeting date, location, and duration).

Although there have been a number of other attempts to develop learning apprentice systems (e.g., (Mitchell et al., 1985), (Segre, 1988), (Schlimmer, 1990), (Kodratoff & Tecuci, 1986), (Nakauchi et al., 1991), (Kozierok & Maes, 1992)) no previous system has passed the threshold of successful deployment within a real user community. We present here preliminary results from field tests of the CAP system by its first regular user.

The following section briefly introduces the calendar management problem. Subsequent sections describe the program and its learning methods, and summarize the learning experiments performed to date. The final section considers lessons learned from this case study and directions for future research.

2. The Calendar Management Task

Our long term objective is to develop a personal calendar agent that provides assistance similar to that of a secretary who has worked for some time with the calendar user, and thus knows the constraints and preferences of that person and the environment in which they work. Such a secretary makes many decisions. Consider, for example, the questions that George's secretary might consider when Dan asks for a meeting with George:

- When should the meeting be held?
- How long should the meeting last?
- At what location should the meeting be held?
- Should an existing meeting be moved to make room for this meeting?
- Should I delay this meeting until next week, or shorten it to make room for it this week?
- Should I ask George whether he wants to meet with Dan, or schedule this myself?
- Should I send a reminder message to Dan (and if so, when)?

The above questions characterize the range of knowledge needed to assist in calendar management. The current prototype CAP system learns to provide advice regarding only the first three types of questions. The other questions represent our goals for the next generation system.

3. The Program

CAP provides an interactive editor for adding, moving, copying and deleting calendar meetings, and for displaying various views of the calendar. CAP is interfaced to the electronic mail system and provides commands for sending automatically generated reminder messages to meeting attendees. It contains a knowledge base that summarizes the current calendar state, information on approximately 250 people and groups known to the calendar owner (including their positions, departments, offices, and email addresses), descriptions of past user commands (which serve as training data for learning), and an evolving set of rules and neural networks for providing advice to the user. The CAP user interface is based on a set of editing commands (macros) written in a text editor called Gnu-Emacs. These editing commands communicate with a Common Lisp subprocess running THEO (Mitchell et al., 1991), a framework for learning and problem solving, on which the core of the system is based. THEO provides the frame-based representation for CAP's knowledge base, inference methods for generating advice to the user, and CAP's learning methods. A more detailed description of a somewhat earlier version of CAP is given in (Jourdan et al., 1991).

Figure 3-1 shows the user interface to CAP, which displays a single calendar week. The user has invoked the Add-Meeting command, and the system is therefore prompting the user for the parameters associated with this command: Event-Type, Attendees, Date, Duration, Start-Time, Location, and Confirmed?. At the bottom of the screen in the figure, the system has prompted for the parameter Duration and has suggested the value of 60 minutes. The user has overridden this value by entering the value 30. *Every* such parameter value for *every* command is both an opportunity for CAP to provide advice to the user, and an opportunity to collect a training example for refining the knowledge upon which system advice is based.

Table 3-1 shows a typical training example captured by CAP as a result of the user adding a meeting to the calendar. The first seven attributes of this training example correspond to the seven basic command parameters which the user must supply (or at least approve). The remaining attributes are inferred upon demand by CAP from the given parameters together with its background knowledge and the current date and time. Note that the possible set of training example attributes includes in principle many more attributes than those shown in Table 3-1. For example, the entire calendar state

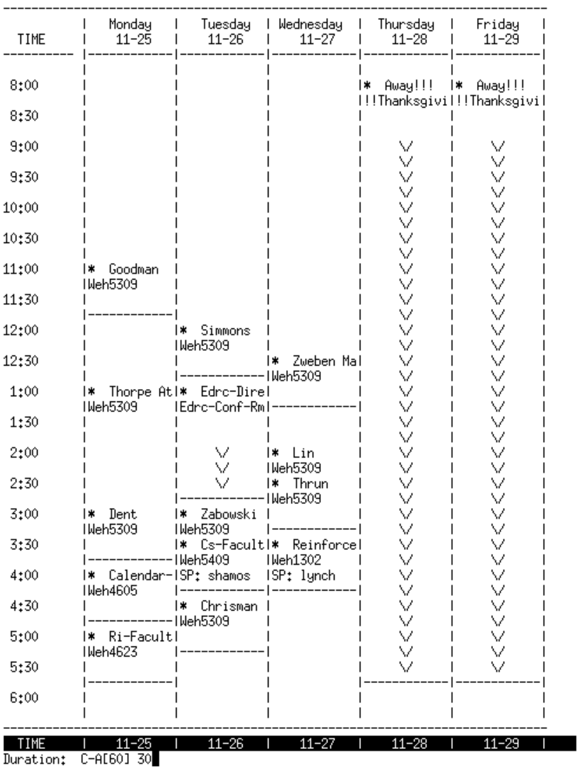

Figure 3-1: A Typical CAP Display.

```
add-meeting-request-217:
     event-type              meeting
     attendees               goodman
     date                    (25 11 1991)
     duration                60
     start-time              1100
     location                weh5309
     confirmed               yes

     action-time             2899823798
     action-date             (22 11 1991)
     day-of-week             Monday
     number-of-attendees     1
     single-attendee?        yes
     lunchtime?              no
     end-time                1200
     attendees-known?        yes
     position-of-attendees   faculty
     dept-of-attendees       gsia
     group-attendees?        no
     sponsor-of-attendees    unknown
```

Table 3-1: A Typical CAP Training Example.

at the time the meeting is added is potentially relevant (including features such as the previous meeting with the same attendees, and other meetings during the same week). Note also that when the user enters a parameter, noise may appear in the resulting training example due to misspellings, synonyms or misunderstanding of the intended semantics of the parameter (e.g. use of both weh5309 and 5309 as synonymous locations). In order to minimize this, CAP enforces an evolving set of legal values during type-in of most parameters, and provides automatic completion where possible.

CAP's learning component makes use of these automatically logged training examples. Each parameter of each command provides a separate learning task (e.g., for the AddMeeting command, learning tasks include learning to predict Duration, Location, and Day-of-Week). The learning component is invoked offline, to avoid delay in user response time. CAP currently utilizes two competing learning methods which are provided by the THEO framework: a decision tree learning method similar to ID3 (Quinlan, 1986, Quinlan, 1987), and a neural network learning method, Backpropagation (Rumelhart et al., 1986). CAP converts each learned decision tree into a set of rules (one rule per path through the tree) which are then further generalized by removing any preconditions that do not increase coverage of negative training examples. In neural network learning, the symbolic training example attributes are automatically encoded as network unit values using a 1-of-n encoding (i.e., an attribute with n possible values is encoded using n network units, one of which will have the value 1 and the others the value 0).

```
If   The Position-of-attendees of event ?x is
         grad-student, and
     The Department-attendees of event ?x is
         robotics;
Then The Location of event ?x is weh5309.

If   The Seminar-type of event ?x is theo;
Then The Location of event ?x is weh4605.

If   The Seminar-type of event ?x is
         distinguished-lecturer;
Then The Duration of event ?x is 90.

If   The Department-of-attendees of event ?x is
         computer-science, and
     The Position-of-attendees of event ?x is
         faculty;
Then The Duration of event ?x is 60.

If   The Attendees of event ?x is
         Mr. Boticario, and
     The Single-attendee? of event ?x is yes;
Then The Day-of-week of event ?x is friday.

If   The Seminar-type of event ?x is theo;
Then The Day-of-week of event ?x is monday.
```

Table 3-2: Some Rules Learned by CAP.

Table 3-2 shows a typical subset of rules learned by

CAP for predicting meeting Duration, Location, and Day-of-Week. Each of these rules is fairly simple, and fairly mundane, but captures some important regularity or preference within the domain. In the current system, CAP has learned a set of approximately 50 such rules for each of its predicted attributes. Some rules cover only a single training example, whereas others cover several dozen examples.

4. Experimental Results and Evaluation

There are several questions to be asked about learning apprentice systems such as CAP. How long does it take to become a useful assistant? How often does the user accept the apprentice's advice? How does the quality of learned advice compare with the quality that could be provided by hand crafted rules? How many features (attributes) need to be considered to produce useful advice? Can learning keep pace with changes that occur over time in a dynamic environment?

Our initial experimentation has been on the prediction of parameters for the Add-Meeting command, specifically prediction of meeting Location, Duration, and Day-of-Week. Briefly stated, our results suggest the following answers to the above questions: CAP can begin to provide useful advice within a month or two. The user's acceptance of advice regarding meeting Location and Duration varies from 45% to 70% over time, but can be significantly improved by suggesting advice only in cases where the apprentice has high confidence. The quality of the apprentice's advice is comparable to the advice we were able to produce with hand-coded rules. The apprentice was able to induce its rules by attending to 10-15 features of meetings and meeting attendees. And the pace of learning is fast enough to capture semester-long regularities in our environment.

Figure 4-1 compares the performance of hand-coded rules for meeting Location against the learned rules and learned neural networks, as the number of training examples is increased. We developed hand-coded rules for each of the parameters of the Add-Meeting command before any learning experiments were conducted, based on our knowledge of the domain and examination of old calendars, at a cost of several person-weeks of effort. The horizontal axis indicates the number of training examples used (the n most recent examples immediately preceding September 30, 1991). The vertical axis indicates the percent agreement between this learned advice and the user for the 60 Add-Meeting commands immediately following September 30. The results of this experiment are typical of those for other data sets and parameters in that they indicate (1) that automatically learned rules provide advice comparable to more costly manually developed rules in this task, (2) that both ID3 and Backprop are capable of learning such knowledge, and (3) that a minimum of 100 training examples should be provided for this learning task.

Figure 4-2 gives more detailed results for predicting

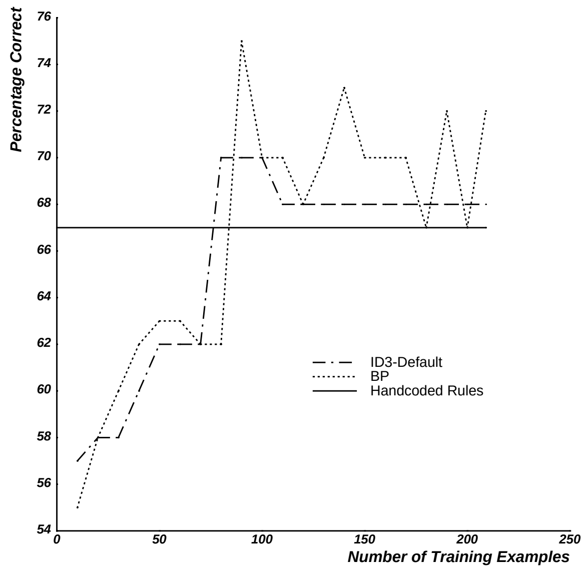

Figure 4-1: Prediction Accuracy versus Training Set Size for Location

Location using 120 training examples. The leftmost bar gives the performance of a simple Default method which always suggests the most common Location value. This shows that 58% of the test meetings for Sept. 30 take place in the default location (the office of the calendar owner). The remaining 42% of meetings are split among 13 additional locations. The bars marked Hand-coded and BP indicate performance of the manually developed rules and the neural network respectively. The method labeled ID3-Default applies the ID3 rules to classify the example, or the Default prediction in the case that no ID3 rule applies[4]. In order to understand the intrinsic difficulty of prediction on this data, we attempted to find a bound on the maximum prediction accuracy that was possible for this set of training and test examples. The bar marked Upper Bound indicates the level of prediction accuracy that can be achieved by assuming that any test example with a previously unseen value for Location cannot possibly be correctly predicted, that test examples with identical descriptions but different classifications will be classified at best using the most common Location value, and that all other test examples will be correctly classified. Notice that the learned classification methods capture approximately half of the predictable cases beyond those with default values.

The lightly shaded bar in Figure 4-2 illustrates the potential to achieve higher classification accuracy by reducing coverage of the data. This ability to trade

[4]In this particular data set, the ID3 rules cover all default predictions, so the Default method does not contribute any successful predictions here.

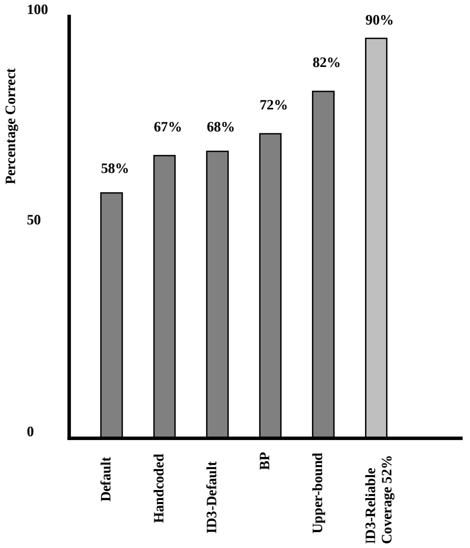

Figure 4-2: Prediction of Meeting Location for meetings following Sept. 30, 1991

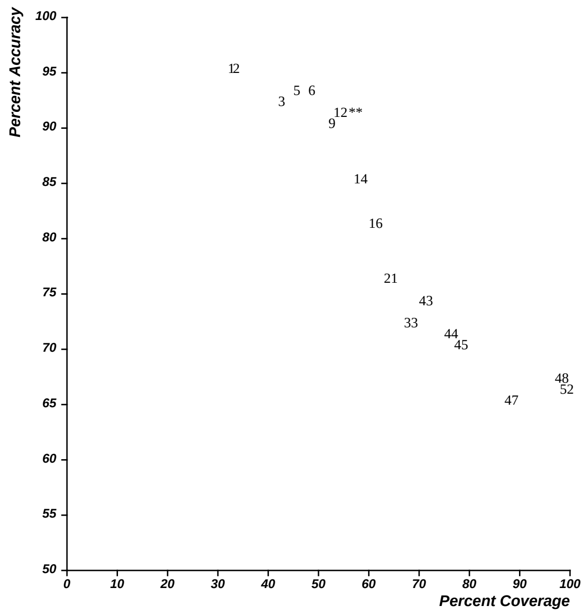

Figure 4-3: Accuracy vs. Coverage for Prediction of Meeting Location for meetings following Sept. 30, 1991

coverage for accuracy is important to the prospects for learning apprentices such as CAP, because it indicates that even in domains where learned knowledge is not strong enough to support reliable decision making in all cases, the agent might still act autonomously in a predictable subset of its domain. One could imagine, for example, a future version of CAP that responds autonomously to email requests for meetings, but only in those cases where the request is covered by highly reliable rules (leaving other decisions to the user). In the figure, the bar labeled ID3-Reliable uses only a subset of the learned rules that CAP estimates to be most reliable. CAP currently judges a rule to be "reliable" if it covers at least two positive training examples, and no negative training examples (the rules given in Table 3-2 are all reliable rules). Using this subset (only 13 of the 53 learned rules), the coverage drops to 52% of the test cases but accuracy over this subset increases to 92%.

The tradeoff of coverage for accuracy for ID3's prediction of meeting Location is depicted in greater detail in Figure 4-3. After the rules are learned using 120 training examples, they are ranked according to their performance on these same examples. The data is sufficiently noisy that many rules do not achieve 100% accuracy even on the training examples. If two rules are equally accurate, the rules which covers more examples is ranked higher. Subsets of the ordered rules are then tested on the 60 examples following September 30. The top rule is tested alone and its performance is plotted as "1" in Figure 4-3, the first and second rule are tested together and plotted as "2", the first three rules are tested and plotted as "3", and so on. In cases where a rule does not apply to any of the test examples, its subset has the same accuracy and coverage as the previous subset (without the inapplicable rule). These subsets are not plotted. For example, no "4" appears on the graph because the fourth rule does not apply to any of the test examples. The subset of rules chosen as "reliable" and reported on Figure 4-2 is marked on Figure 4-3 by a double star.

Figure 4-3 shows that CAP has the ability to rank its rules and to increase the accuracy of its predictions of meeting Location by selecting a subset of reliable rules. Similar tradeoffs of coverage for accuracy have been obtained when predicting Duration and Day-of-Week, though the absolute accuracies tend to be lower for these attributes. The same tradeoff could also be used on the Backpropagation predictions (by using the real-valued output activations as measures of confidence), but we have not yet done so.

Whereas the above data is based on a single set of

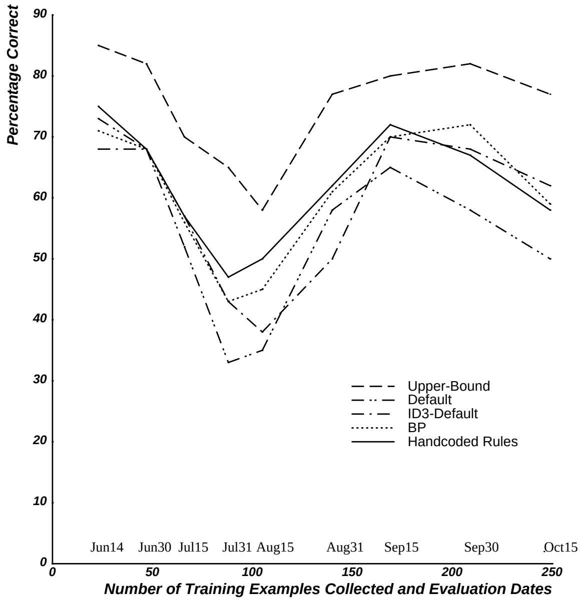

Figure 4-4: Prediction of Meeting Location with 100% Coverage

training and test data surrounding September 30, 1991, the data illustrated in Figure 4-4 summarizes the evolution of system performance due to learning over the interval from June 1991 through October 1991[5]. For each of the dates indicated on the horizontal axis, ID3 and Backpropagation were invoked using a training example window of up to 120 meetings immediately preceding that date. Performance of the learned classifiers was then evaluated using the next 60 Add-Meeting commands following that date. The five lines on the graph indicate the classification accuracy of the first five methods on the bar graph of Figure 4-2. All methods here are evaluated relative to 100% coverage of the test data (the tail of the curve in Figure 4-3).

Several trends are apparent from Figure 4-4. First, consider the overall shape of the curves. In spite of the wide variance in predictive methods depicted, they all follow the same general pattern; a pattern dictated by the data which is being tested. The low points during July and August are likely due to differences in the distribution of calendar meetings between the academic semester (which begins September 1) and the summer break. During the semester there are strong regularities in meetings, whereas during the summer the schedule becomes much less structured, and prediction becomes correspondingly more difficult. Other features of the data are less easily explained, such as the high predictability of June meetings. In the remainder of this discussion, we will concentrate on the relative performance of the different methods over this varying data.

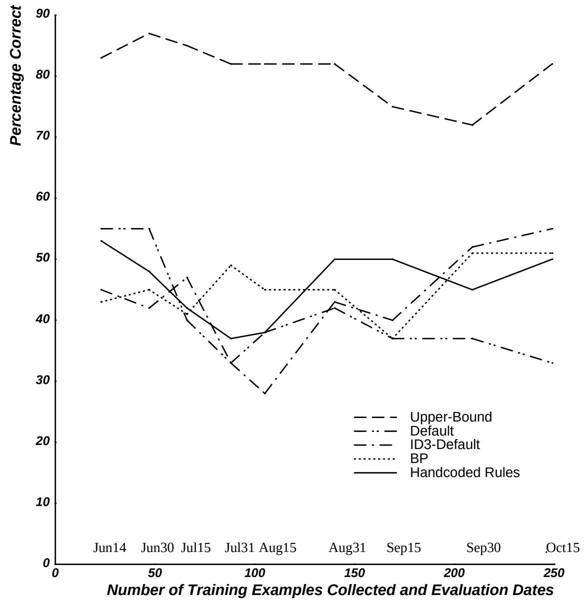

Figure 4-5: Prediction of Meeting Duration with 100% Coverage

Consider the performance of the learned rules and the neural network. Note that the training window does not reach 100 examples until approximately August 15, so performance of learned knowledge before this date is hindered by insufficient data. Following this date, two trends seem to emerge in this data and the corresponding graph for meeting Duration shown in Figure 4-5. First, notice that the date on which the worst performance of learned knowledge is found *relative to the Default and Upper-Bound* is approximately August 31. At this point, the window of training examples is completely filled with examples from summer break, whereas the test examples (following August 31) are meetings from the fall semester. Thus the especially poor performance in this case may be due to a training set that exhibits regularities that are misleading relative to the test set.

Second, notice that performance of the learned classifiers (for both Location and Duration) change from somewhat below the hand coded rules to somewhat above the hand coded rules as the semester continues. Two factors may help to explain this trend. First, the window of training examples gradually fills with fall semester meetings rather than summer meetings, so that learned knowledge becomes more correct as the semester wears on. Second, as time goes on, permanent changes in the environment (e.g., new meeting rooms, new committees)

[5]Although learning continues up through the present time, October is the latest date for which 60 subsequent test examples are available to evaluate the quality of learning.

gradually cause the hand coded rules (written in April, 1991) to become out of date.

Based on the data obtained so far, we predict that in the future the performance of the learning methods will remain around 65% correct for prediction of meeting Location and 50% correct for prediction of meeting Duration; lower between semesters and over the summer. We may be able to improve these results by adding new input features to the learning methods. For example, the relative fullness of the day or week of a meeting may influence its duration, but this feature is not currently considered by CAP's learning methods. It may also help to make more features observable to CAP. For example, if the availability of conference rooms were accessible online, it would be a valuable feature for predicting meeting Location. Another promising direction is to target the prediction of a subset of the meetings by using the coverage for accuracy tradeoff discussed earlier. There are several variations of this technique which we intend to explore. Currently CAP learns a completely new set of rules for each date and selects the most reliable rules from these, as measured over the training examples. We plan to extend this by saving reliable rules between learning episodes, enabling the set of reliable rules to grow. We also intend to experiment with different definitions of "reliable" and different methods of sorting the rules.

5. Summary and Conclusions

Our experience with CAP provides preliminary empirical support for our thesis that personal learning apprentices can produce useful knowledge-based assistants with significantly reduced manual development and maintenance of the knowledge base. CAP demonstrates that it is possible to design a learning apprentice which is attractive to users, able to capture useful training data, and capable of generalizing from such data to learn a customized knowledge base competitive with hand coded knowledge. Lessons learned from our experience with CAP thus far include the following:

CAP is able to collect useful training data for several reasons. First, it is attractive to users even before it has learned any knowledge, because it provides a convenient set of calendar editing and display commands and an interface to the email system. Second, it is able to collect the training examples it requires without imposing a significant extra burden on the user. Third, the interface is designed to maximize consistency in the data it obtains by providing automatic completion and checking of legal values.

Some manual updating is still required, even though learning reduces the problem of maintaining the knowledge base. In particular, CAP depends on knowledge about meeting attendees, and as new attendees are encountered this knowledge must be updated. In the current version of the system, the user is prompted for the institution, department, position, and email address of each new attendee who is not yet described in its knowledge base. The system has now built up a knowledge base of about 250 individuals. We are currently considering interfacing the system to an online personnel database to reduce this load on the user.

Selection of predictive attributes for each predicted parameter is a critical part of system development. Predictive attributes must be relevant, observable, and efficiently computable. Research on learning mechanisms that could handle larger numbers of attributes could ease the task of manual selection of these attributes. We conjecture that the set of attributes we select for each learning task in the calendar system will be stable enough across different users and across time to produce useful personal assistants.

There are many features that influence meeting Duration, Location, etc. that are unobservable to CAP. For example, meeting Location is influenced by the availability of conference rooms at the time of the meeting, and meeting date is influenced by the calendars of other attendees. Note that the observability of the world could be increased if other users in our environment start to use the system to maintain their calendars, or if CAP were able to access online information about conference room schedules.

Learned regularities in the environment change over time. For example, meetings tended to be longer and fewer during the summer than during the academic semester, and new committees with new meeting locations appeared. In order to succeed in the face of a drifting environment, the time constant of the learner must be shorter than the time constant of the environment. A window of 120 examples, which corresponds to a few months worth of data from our current user, provides sufficient data for predicting meeting Location and Duration, given the 10 or so features we use to predict each. New learning techniques that reduce the number of required training examples would be a great help.

It is possible to trade off completeness for improved accuracy in the system's advice. For example, by using only its most reliable rules, CAP is able to increase its accuracy for predicting Location to 92% at the cost of reducing its coverage to only 52% of the examples. The ability to make this tradeoff opens the opportunity for highly reliable autonomous operation (e.g., for responding automatically to meeting requests when the user is absent) in which the system responds only to cases for which it believes its response is most reliable.

In addition to the research issues raised above, extensions to CAP that we hope to explore in the future include (1) giving copies to multiple users to determine CAP's ability to customize to each, (2) exploring the opportunities for collaboration and co-learning among several copies of the system, (3) adding the ability to provide pro-active advice (e.g., to suggest that a reminder message should be sent, rather than waiting for the user to initiate the action), and (4) extending CAP to respond automatically to email requests for meetings.

6. Acknowledgements

We wish to thank Jean Jourdan for helping develop the initial version of CAP, and Alan Christiansen, Lonnie Chrisman and Sebastian Thrun for providing helpful comments on this paper. We also thank the entire THEO group at CMU for providing the framework in which CAP is written.

This research was sponsored in part by the Avionics Lab, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597 and by a grant from Digital Equipment Corporation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government or of Digital Equipment Corporation.

References

Jourdan, J., Dent, L., McDermott, J., Mitchell, T., and Zabowski, D. (1991). *Interfaces that Learn: A Learning Apprentice for Calendar Management* (Tech. Rep.) CMU-CS-91-135. Carnegie Mellon University.

Kodratoff, Y., and Tecuci, G. (1986). *DISCIPLE : An Iterative Approach to Learning Apprentice Systems* (Tech. Rep.) UPS-293. Laboratoire de Recherche en Informatique, Universite de PARIS-SUD.

Kozierok, R. & Maes, P. (1992). Intelligent Groupware for Scheduling Meetings. *Submitted to CSCW-92*. .

Mitchell, T.M., Mahadevan, S., and L. Steinberg. (August 1985). LEAP: A Learning Apprentice for VLSI Design. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*. , Morgan Kaufmann.

Mitchell, T. M., J. Allen, P. Chalasani, J. Cheng, O. Etzioni, M. Ringuette, and J. Schlimmer. (1991). Theo: A Framework for Self-improving Systems. In VanLehn, K. (Ed.), *Architectures for Intelligence*. Erlbaum.

Nakauchi, Y., Okada, T., and Anzai, Y. (May 1991). Groupware that Learns. *Proceedings of the IEEE Pacific Rim Communications, Computers and Signal Processing*. , IEEE Press.

Quinlan, J.R. (1986). Induction of Decision Trees. *Machine Learning*, *1*(1), 81-106.

Quinlan, J.R. (August 1987). Generating Production Rules from Decision Trees. *Proceedings of the International Joint Conference on Artificial Intelligence*. , Morgan Kaufmann.

Rumelhart,D. E.,G. E. Hinton, & R. J. Williams. (1986). Learning Internal Representations by Error Propagation. In Rumelhart, D. E., G. L. McClelland, & the PDP Research Group (Ed.), *Parallel Distributed Processing.*. MIT Press.

Schlimmer, J.C. (November 1990). Redstone: A Learning Apprentice of Text Transformations for Cataloging. Unpublished.

Segre, A.M. (1988). *Machine Learning of Robot Assembly Plans*. Kluwer Academic Press.