An Adaptive Stock Tracker for Personalized Trading Advice

Jungsoon Yoo
Computer Science Department
Middle Tennessee State University
Murfreesboro, TN 37132 USA
csyoojp@mtsu.edu

Melinda Gervasio and Pat Langley
Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, CA 94306
{gervasio, langley}@isle.org

ABSTRACT

The Stock Tracker is an adaptive recommendation system for trading stocks that automatically acquires content-based models of user preferences to tailor its buy and sell advice. The system incorporates an efficient algorithm that exploits the fixed structure of user models and relies on unobtrusive data-gathering techniques. In this paper, we describe our approach to personalized recommendation and its implementation in this domain. We also discuss experiments that evaluate the system's behavior on both human subjects and synthetic users. The results suggest that the Stock Tracker can rapidly adapt its advice to different types of users.

Categories and Subject Descriptors

H.5 [**Information Systems Applications**]: Information Interfaces and Presentation

General Terms

Design, experimentation, human factors

Keywords

Adaptive user interfaces, machine learning, user modeling, personalization, information filtering

1. INTRODUCTION

With the advent of the Internet, a wealth of information awaits anyone within the touch of a few keystrokes. Unfortunately, the desired content is often buried in massive amounts of irrelevant information and each user must cull through the extraneous material. For example, online brokerage firms now let one check stock prices and make transactions through a Web browser. With all the stocks available throughout the world, there are more opportunities to make or lose money, but only if one has the time and energy to follow those stocks. Tracking tens of thousands of stocks is beyond the capability of any single user.

IUI'03, January 12–15, 2003, Miami, Florida, USA.

Information filtering systems address the problem of information overload by factoring out irrelevant content and reducing the amount of information that the user must examine. Such systems could give an investor more opportunities to examine potentially profitable stocks by eliminating obviously nonprofitable ones. This task of separating interesting from uninteresting information can be viewed as a classification task. However, since different people have individual tastes, information filtering should be personalized. This can be achieved through user models or profiles that embody the preferences of a user or group of users. Moreover, such profiles can be learned from traces of interactions with individual users.

In this paper, we describe the Stock Tracker, an adaptive user interface that recommends stocks based on an individual's trading profile. The system utilizes this profile to rank stocks, and it revises the profile based on traces of user behavior. We have evaluated this prototype experimentally on historical records from the S&P 500 stock market and obtained positive preliminary results. In the next section, we define the stock tracking problem formally. We then discuss our technical approach, including the overall system architecture, engineering decisions, and implementation issues. After this, we present some hypotheses about the Stock Tracker's ability to adapt to individual users and describe the systematic experiments we ran to test them. In closing, we review related work and consider some directions for future research.

2. THE PROBLEM OF STOCK TRACKING

Through the Internet, ordinary people can now access nearly 3500 companies that make up the New York Stock Exchange, almost 4500 securities in the NASDAQ composite, and many more stocks are available through various Web sites. The values of these stocks change continuously, while news and trading information regarding a company's financial status are available instantly. This makes it virtually impossible for anyone to keep a close watch on more than a handful of stocks.

Many commercial Web sites partially address this problem by presenting buy recommendations on a small set of stocks. However, few of them make complementary sell recommendations, as this would require knowledge of each user's portfolio. Furthermore, different traders have different investment styles, in that some are aggressive risk takers and others are more concerned with long-term returns. Existing sites do not typically cater to such individual preferences.

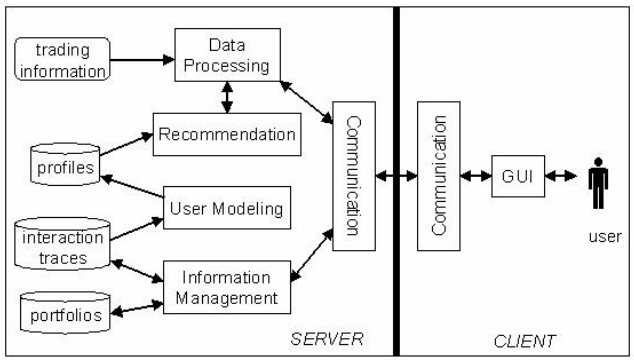

Figure 1: Architecture for the Stock Tracker.

Our goal is to build an adaptive stock tracking system that acquires an individualized profile automatically through interaction with a user, then utilizes this profile to generate personalized trading advice. Adaptive stock tracking can be viewed as a specialized case of the more generic task of filtering events in a high-dimensional space. Given many continuous variables that change over time and online data for these variables sampled at regular intervals, the information filtering task is to find trends or events in the data that the user will find significant.

What is "significant" depends on the specifics of the problem domain. For example, if the task involves monitoring events in a complex assembly plant, a significant event may be a divergence from planned developments that reveals the possibility of an accident. Significance can also depend on the user, which suggests a clear role for personalization. For instance, depending on a user's expertise, an information filtering system might alert the user at different times or present information at different levels. By automatically adapting its behavior to individual needs and preferences, an information filter can further increase its utility as an assistant in complex temporal domains.

We have built a prototype stock tracking system that learns personal preferences cast within a decision framework based on a pure technical analysis [1]. That is, the analysis uses only temporal stock trading data (e.g., stock price, trade volume), and not any financial information about the company. Thus, we can state the stock tracking problem formally as a temporal information filtering task:

Given: A user and his associated profile;

Given: A large set of stocks available to buy and sell;

Given: Online trading information for these stocks collected at regular intervals; and

Given: Strategies for how to make a trading decision;

Find: A list of trading recommendations in an order that reflects the user's priorities.

To make personalized recommendations, the Stock Tracker relies on user profiles that embody individual preferences. These profiles are acquired automatically by calling machine learning techniques on user interaction traces. The system gathers data unobtrusively, taking advantage of an interface design that obtains useful information from a user's natural interactions. In the next section, we present the details of our approach to adaptive stock tracking.

3. AN APPROACH TO STOCK TRACKING

To reiterate, the Stock Tracker addresses the problem of information overload through a personalized approach to information filtering. By considering personal user preferences, the system filters trading information in an individualized manner, presenting only information that a user finds interesting. In this section, we present the details of the Stock Tracker. After presenting the architecture, we elaborate on the graphical interface, the recommendation module, and the user modeling component.

3.1 System Architecture

The Stock Tracker is built on a client-server architecture, with information filtering, record keeping, and adaptation performed on the server, while the user interface and related computing are done on the client, as Figure 1 depicts. The server contains the data processing unit, recommendation module, user modeler, information manager, and communication unit. The data processing unit converts raw input (i.e., current stock readings and historical trading information) into reports that contain buy and sell recommendations for the user. It relies on the recommendation module to make appropriate suggestions for each stock based on individual user profiles. The user modeler constructs these profiles based on user responses to previous recommendations. The information manager records traces of a user's interactions with the system and also keeps track of user portfolios. Finally, the communication unit manages the information into and out of the server.

A client contains a communication unit and a graphical user interface (GUI) component. The communication unit performs activities that correspond to the server's communication unit. Meanwhile, the GUI presents all reports to the user and accepts commands such as buying/selling stocks and viewing portfolios, along with requests for additional financial information on particular companies. The system simulates trading using historical S&P 500 market data; it mimics a real stock trading scenario by generating information one (simulated) day at a time and letting a trader decide the stocks, if any, to buy and/or sell on each day.

3.2 Interface for Unobtrusive Data Collection

The Stock Tracker includes a graphical interface, shown in Figure 2, for presenting stock information, making recommendations, and accepting the user's trading requests. The system's ranked list of recommendations appears in the upper left. Details about the highlighted stock are presented in the interactive graph in the bottom half of the window. The upper right presents a summary of the current stock, together with the system's recommendation and action buttons for the user to buy or sell the stock. The user may also, but need not, provide additional feedback on the recommendation, such as clicking on "Thank you" to indicate agreement with a warning or specifying a desired alternative in the case of an unacceptable recommendation.

Although we have not systematically tested this graphical interface for usability, we have tried to adhere to known principles, such as Shneiderman's [20] eight golden rules of interface design and Karat's [7] principles of usability. Since the system alters its advice based on interactions with the user, we were also particularly concerned about its methods for obtaining user feedback. Some approaches to personalization require users to state their preferences explic-

Figure 2: Graphical interface of the Stock Tracker.

itly through long questionnaires or indicate directly whether each recommendation is good or bad. Instead, we prefer unobtrusive data collection techniques and have designed the Stock Tracker's interface so that it can obtain useful feedback through a user's natural interaction. For example, a user can provide positive feedback by purchasing a stock that the system recommends he should buy. By selling the same stock, the trader gives negative feedback. Because more explicit feedback is also helpful, we also provide this facility, but the Stock Tracker can adapt its behavior to users even without such information.

3.3 Recommending Trading Actions

At the heart of the Stock Tracker are the personalized trading recommendations that it makes to each user. The system bases these recommendations on a technical analysis called Moving Average Convergence Divergence (MACD) [1] that examines the difference between long-term and short-term moving averages to identify crossing points. These points indicate market turns and thus correspond to opportunities for buying or selling stock. Specifically, the time to buy stock is when both the long-term and short-term averages are increasing and the short-term average has exceeded the long-term average. Conversely, the time to sell is when the averages are decreasing and the short-term average has dipped lower than the long-term average. We compute short-term averages over nine days and long-term averages over 18 days. We use MACD because it is a simple technique that is still popular in stock market analysis.

We convert MACD into decision rules for recommending five different actions: *buy*, *buy warning*, *sell*, *sell warning*, and *do nothing*. The *buy* and *sell* rules correspond directly to those just described. A *buy warning* occurs when a recommendation to *buy* is likely in the near future—that is, both averages are increasing but the short-term average remains lower than the long-term average. Similar logic applies to the *sell warning*, while all other situations correspond to *do nothing*. Although MACD specifies the conditions for each action, the Stock Tracker must still determine exactly when to make each recommendation with respect to when the short-term average crosses the long-term average. Different choices correspond to different risk levels and capture different preferences about how soon to give buy and sell warnings. The MACD technique gives, in some sense, objectively optimal buy and sell recommendations, but our primary objective in building an information filtering assistant is to help users make their own decisions more efficiently, rather than to make decisions for them.

Each decision rule consists of a set of numeric constraints on temporal stock-trading attributes, such as the rate of increase in the long-term moving average or the difference between the long-term and short-term averages. A decision rule applies when all of its constraints are satisfied—that is, the value of each corresponding attribute satisfies the constraint. For example, we can encode "the long-term average is increasing" with the constraint (LTA_slope $\geq \alpha$). By using a parameter α instead of zero, we allow for differences in how much a stock price must be increasing for an individual to consider it. Thus, while MACD defines the form of the constraints, the Stock Tracker can alter its recommendation behavior by incorporating different threshold values.

3.4 Personalizing Stock Recommendation

The Stock Tracker achieves personalized recommendation through the use of individual user profiles that capture trading preferences. A profile consists of four binary classifiers, one for every action other than *do nothing*, each of which renders a membership decision on each item (i.e., whether it is a positive instance of the class). In the Stock Tracker, each binary classifier embodies a MACD decision rule and makes a recommendation if its constraints are satisfied. For each stock, the recommendation module selects the recommendation with the highest associated confidence value, or *do nothing* if no classifier recommends an action. This confidence value is then used to present the selected recommendations on different stocks in a ranked list.

The system builds classifiers from training examples extracted from traces of the user's interactions. Briefly, the user can either accept or reject each recommendation. An acceptance indicates that the recommendation was correct and is thus a positive example of the corresponding classifier. Similarly, a rejection produces a negative example. The system also uses positive examples for one classifier as negative examples for others. Although there exist many well-known supervised induction algorithms, we opted to devise a new, more efficient algorithm that exploits the fixed structure of the user model, as Table 1 summarizes.

Recall that each decision rule (classifier) consists of a set of constraints, each of which corresponds to a particular numeric attribute. Every constraint specifies a threshold on the attribute value, above (below) which the constraint is satisfied. The goal of the learning algorithm is thus to find a set of thresholds that will result in recommendations consistent with the user's actions. If we order the examples according to increasing attribute values, we can evaluate candidate thresholds for a $\geq$ constraint based on how well they predict positive examples above the threshold and negative examples below it, and similarly for $\leq$ constraints. Also, since the constraints form a conjunctive set of conditions,

Table 1: Learning algorithm for the Stock Tracker

Learn(*examples*)
For each *classifier* in {buy,buy warning,sell,sell warning}:
LearnOne(*classifier*, *examples*, *classifier*'s constraints)

LearnOne(*classifier*, *examples*, *constraints*)

- Sort *examples* in increasing order according to the attribute value of the next *constraint* in *constraints*.
- Identify threshold *candidates* for splitting the *examples* into positive and negative regions.
- Set *constraint* threshold of *classifier* to the best *split* among *candidates* based on Evaluate(*examples*, *split*).
- LearnOne(*classifier*, Subset(*examples*, *split*), remaining *constraints*).

thresholds for remaining constraints need only be considered for examples in the region that satisfies the constraint.

We tried a variety of methods for evaluating candidate splits, including information gain and various metrics based on precision and recall, which are often used to evaluate information retrieval systems. We found the F measure [14], a weighted combination of precision and recall, to provide the best behavior. Precision indicates the probability that a positive instance labeled as positive by the classifier is truly positive, whereas recall gives the probability of correctly identifying all positive instances. In our case, precision is the number of positive examples in the correct partition divided by the number of instances in that partition, while recall is the number of positive instances in the correct partition divided by the number of positives in both partitions.

A primary motivation for the development of an efficient algorithm is that learning is conducted online; that is, after every interaction with the user that yields positive or negative examples, the user modeler updates the user profile. This lets the Stock Tracker adapt quickly to individual traders. To give the system a reasonable starting point, we employ a default model, corresponding to the profile of an average user, that it induces from a default training set that represents feedback from such a user. By attaching a weight to these default cases, we can vary the degree to which the system relies on them during its model construction.

4. EXPERIMENTAL EVALUATION

Our goal in developing the Stock Tracker was to help users identify more quickly stocks to buy or sell based on their individual interests. By adapting its model of the user's trading preferences based on interaction traces, the system tailors its recommendations. As the Stock Tracker gains experience with a user, it should become better at recommending actions that he will accept. Here, we describe the results of experiments conducted with human and synthetic subjects that we designed to test this primary hypothesis.

4.1 An Experiment with Human Subjects

The basic question we want to answer is whether the Stock Tracker can adapt its recommendations to different users. We can evaluate the system's performance by measuring its success in recommending trading actions about various stocks that users find acceptable. Specifically, for each individual, we can measure the percentage of times he accepts the system's recommendations, with higher percentages corresponding to better performance. We can also measure the time taken to complete transactions. If the Stock Tracker makes good recommendations and ranks them highly, the user should execute his transactions rapidly. Thus, we expected the acceptance rate for a given user would gradually increase while the time per transaction would decrease.

To test this hypothesis, we conducted an experiment with twelve human users who had various levels of stock trading expertise. The subjects attended a brief orientation session to learn how to use the system, and we asked them to complete one practice session to familiarize themselves with its operation. The study simulated daily trading based on 2001 S&P data, with each subject interacting with the system for at least 50 transactions or suggestions, but having the option to finish the experiment over multiple sessions. Each user began the study with $20,000 in his portfolio, roughly half of which was already invested in selected stocks. As the subject interacted with the Stock Tracker, we measured the time taken to complete each transaction and the acceptance rate, that is, the number of user actions matching the recommendations divided by the total number of such actions.

Figure 3 shows the acceptance rate as a function of the number of user actions (i.e., training cases). At first glance, the results are somewhat surprising, in that they show an initial decrease in acceptance rate followed by an increase. They are also somewhat disappointing, with the final acceptance rate not going much higher than the initial level. However, analysis of user traces and interviews with subjects suggested that the initial acceptance rate was artificially inflated because new users tended to focus on expanding their portfolio, basically ignoring the initial one provided. They did this mainly by selecting from the Stock Tracker's buy recommendations, which produced the high initial acceptance rate. However, as users began to monitor these stocks, they also started making transactions that differed from the system's recommendations, decreasing the acceptance rate. But as the Stock Tracker gained experience with the user's preferences, it began making recommendations they found acceptable, increasing the score again.

Figure 4 also shows the average amount of time spent on each transaction as a function of the number of user actions, which decreases early on but then levels out. We suspect that this trend is due partly to users becoming more effective at interacting with the Stock Tracker as they gained experience with it, despite their practice before we began collecting trace data. However, the reduction in transaction time is also consistent with the view that the system's adaptation to users improved its ability to rank acceptable recommendations more highly, thus reducing users' effort at finding stocks to buy or sell.

4.2 An Experiment with Synthetic Subjects

The experimental results just described are somewhat ambiguous, and interviews of subjects revealed they often had trouble understanding how to proceed during their early interactions with the Stock Tracker. Thus, it seems likely that the effects of the system's adaptation to inexperienced users were confounded with those subjects learning simultaneously about the investment process themselves. In contrast, we designed the Stock Tracker for knowledgeable traders who have clear preferences about the stocks in which they prefer to invest.

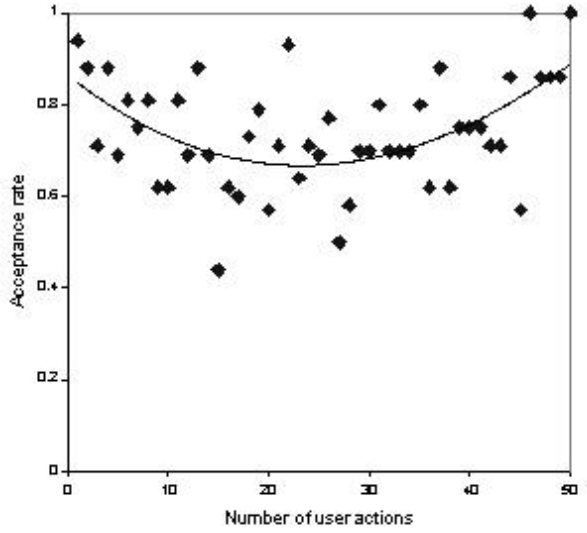

Figure 3: Performance of Stock Tracker with human subjects in terms of acceptance rate.

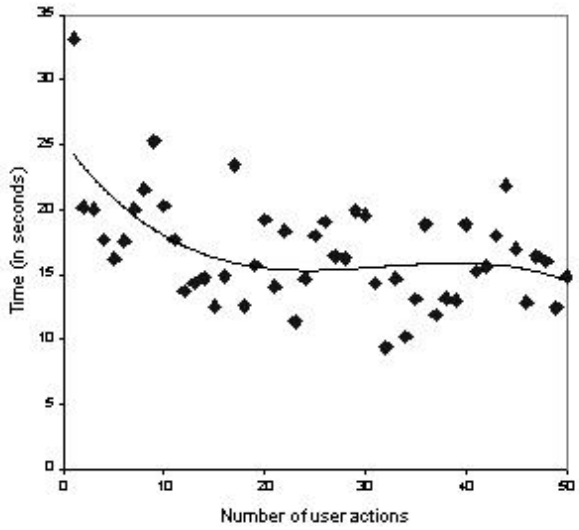

Figure 4: Performance of Stock Tracker with human subjects in terms of time spent on each transaction.

One response would be to repeat our experiment with a more experienced population of users, but we had no ready access to such a subject pool. Moreover, as both Drummond et al. [6] and Gervasio et al. [7] have argued, synthetic subjects offer advantages over human users in that they let one eliminate improvement due to learning by subjects and they generally make it easier to run carefully controlled and repeatable experiments. Thus, we decided to construct a set of such synthetic users to serve as subjects in a second experimental study.

As before, we wanted to determine whether the Stock Tracker can tailor its recommendations to individual users. Recall that the system constructs user models of a fixed form based on the MACD technical analysis. By varying the threshold parameters in the model, we can effectively create users with different investment styles. We can interpret the actions that a given model predicts as the actions made by the corresponding synthetic user. As in the previous experiment, we utilized acceptance rate as the dependent variable to measure success, and we predicted that this performance measure would increase as the Stock Tracker gained experience with each subject. However, our synthetic users did not attempt to mimic latency in making a selection, so we did not measure transaction time in this study.

To test our hypothesis about acceptance rate, our experiment included 200 synthetic investors sampled from a uniform distribution that ranged from very conservative to very aggressive. We repeated the previous experimental setup by testing each user in a simulated online fashion on 2001 S&P data. We started each user at a randomly chosen date from the first 50 days of the year and with the same initial portfolio. We initialized each user model with a default training set, as described earlier. For each day, the Stock Tracker used the current model for a given user to generate a list of recommended transactions, ordered by their associated confidence value. We filtered the list to exclude any sell or sell warning recommendations for stocks the user did not own. To simulate regular user behavior, we then randomly picked for evaluation five of the top ten recommendations plus one other from the rest of the list. We then compared the recommendations of the current model on these stocks to the user's actions (as predicted by the synthetic user model). As before, we measured the system's acceptance rate as the number of matches divided by the total number of actions. Finally, the Stock Tracker updated its model based on the user's actions in preparation for the next day. For each user, we also ran a control condition where no adaptation occurred, in that user interaction did not modify the model.

Figure 5 shows the Stock Tracker's acceptance rate as a function of the number of user actions, averaged over the 200 users. The graph shows that the adaptive version achieves an acceptance rate of about 98% after fewer than 30 user actions (i.e., training examples). In contrast, without learning, the acceptance rate was only about 88%.[1] These results with synthetic subjects support our hypothesis that the Stock tracker can successfully adapt to experienced investors with different trading preferences.

4.3 An Experiment with Default Models

As we have explained, the Stock Tracker initializes user models with a default training set. This is primarily to provide reasonable recommendations even for a new user. However, this reliance on a default model can also affect the Stock Tracker's ability to adapt to users. A crucial issue with adaptive interfaces, particularly ones that learn online, is rapid adaptation from few interactions. Thus, it is important that a default model provide good initial advice without sacrificing the ability to learn individual preferences quickly.

Three questions arise with respect to default models for stock tracking. The first concerns the trading strategy that it should embody. Preliminary studies suggested that a moderate model (i.e., neither too conservative nor too aggressive) offered the best tradeoff between recommendations

[1]The slight upward trend for the nonadaptive condition suggests that the actions for the stock trading situations later in the year were easier to predict for the default model. This could be because the latter part of the year was more similar to the 2000 S&P data on which we based this model.

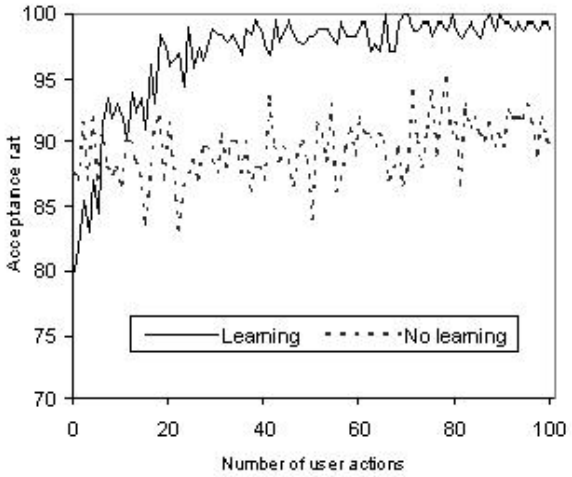

Figure 5: Acceptance rate for Stock Tracker recommendations with and without learning.

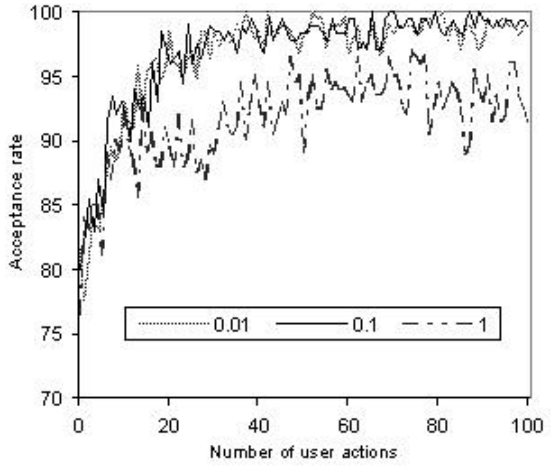

Figure 6: Acceptance rate for Stock Tracker recommendations with different default model weights.

the user would accept and those he would reject, thus giving useful feedback for updating the model. The second issue involves the size of the training set used to construct the default model. We settled on 200 training cases after preliminary experiments showed that this number offers reasonable initial advice for different types of users.

A third question, which we set out to answer more formally, concerns the appropriate weight placed on the default training set. Smaller weights give more importance to the data generated by user interaction; for example, a default weight of 0.1 means that it takes ten default examples to have the same effect as one example based on a user action. For different weight settings, we care about the initial acceptance rate, the asymptotic acceptance rate, and the speed of convergence. We predicted that an intermediate weight on the default model would give the best tradeoff between initial acceptance rate and rapid adaptation.

To test this hypothesis, we created a default training set using 200 randomly chosen instances from the 2000 S&P data, which we labeled with the predictions of a moderate synthetic user. We then ran an experiment with 200 synthetic users, in a simulated online fashion on 2001 S&P data, as in the previous study. We examined three different weights for the default model – 1.0, 0.1, and 0.01 – which produced the results in Figure 6. As expected, the intermediate weight (0.1) gave a better balance between initial and asymptotic behavior than did the high weight (1.0). However, the low weight (0.01) produced a very similar learning curve to the intermediate condition, which we did not anticipate, suggesting the Stock Tracker's behavior is less sensitive to this parameter than predicted.[2]

5. RELATED AND FUTURE WORK

Much of the research on adaptive information filtering has focused on document retrieval or text categorization, especially for applications related to the Internet such as Web browsing [2,15], e-mail processing [18], and news reading [4,11]. Unlike the problems that have been tackled by these approaches, stock tracking is a temporally sensitive task that requires the continuous monitoring of numeric variables to detect trends or changes over time. This requires identifying features that capture these trends and employing techniques to gather such information for the user. Perhaps the closest work in text filtering comes from Languillon [13], who studies the problem of changing content, but his task still involves filtering static documents, rather than items like stocks that are themselves changing over time.

Machine learning approaches to personalized recommendation are often divided into two broad classes. Collaborative approaches [11,19] make recommendations based on user similarity as evidenced by their item ratings, whereas content-based approaches [12] make recommendations based on item similarity as evidenced by their content descriptions. Collaborative methods perform well when there is substantial overlap between the users and the items they have rated, but fare less well for users with idiosyncratic behavior and on items for which few ratings exist. Content-based approaches like the Stock Tracker do not suffer from these limitations, as they learn individual preferences over item attributes. However, they require content descriptions for items and thus are better suited to domains where such information is available. Research on combining collaborative and content-based techniques [2,3] attempts to address the limitations of each and could prove useful in our domain.

Early systems for personalized recommendation often imposed a significant burden on users to provide explicit feedback on system advice. More recent work has proposed solutions ranging from asking the user to rate the most informative items [16], using synthetic users to augment the user pool [17], and using implicit feedback to approximate explicit ratings [5,9]. Rather than use low-level events such as mouse and keyboard tracking to indicate interest or approval, the Stock Tracker relies on the implicit feedback [4,7,12] that comes naturally in the course of the user interacting with the system—accepting recommendations to buy or sell stocks, or executing alternative actions.

The experimental results presented in the previous section provide evidence that the Stock Tracker's can adapt rapidly to users with different investment styles. However, there remain additional issues to investigate. We should replicate

[2]Because it produced reasonable behavior, we relied on the intermediate setting in the experiment on synthetic and human users reported earlier in the section.

the study on experienced synthetic users with human subjects, ensuring that users have a reasonable knowledge of trading. Additional studies with human subjects should examine whether user models based on the MACD framework are sufficient to capture preferences across a wide range of users. We can easily replace the profiles utilized in the Stock Tracker's modeling component with any one of the many technical analysis methods available [1]. We can also design synthetic users for alternative model types and evaluate the Stock Tracker's ability to adapt to them.

An additional open question concerns how to encode and acquire more complex models. One approach would impose a hierarchy over the available stocks, such as that defined in the Global Industry Classification Standard [8]. We could extend the Stock Tracker to adapt to users' preferences for each node in the hierarchy. This approach would capture user preferences for different types of stocks, such as those for software companies vs. automobile companies, but probably at the expense of a slower learning rate than for a simpler user model.

In summary, the Adaptive Stock Tracker is an information filtering system that rapidly tailors its trading recommendations to users with different investment styles. The advisor does this by acquiring a model of user preferences automatically from traces of user interactions, utilizing an efficient algorithm that exploits the fixed structure of the user model. The Stock Tracker does not require users to fill in lengthy questionnaires about their preferences or to provide detailed feedback about its recommendations. Instead, it collects traces in an unobtrusive manner and extracts training instances from its natural interaction with the user. Experimental results with both human and synthetic subjects provide support for this approach, suggesting that the Stock Tracker learns to make increasingly acceptable recommendations as it interacts with individual users.

ACKNOWLEDGEMENTS

The research reported in this paper was supported in part by Grant NCC 2-1220 from NASA Ames Research Center. We thank Stephanie Sage and Daniel Shapiro for their contributions to the formulation of the problem.

REFERENCES

[1] S. B. Achelis. *Technical Analysis From A To Z*. Probus Publishing, Chicago, 1995.

[2] M. Balabanovic and Y. Shoham. Fab: Content-based collaborative recommendation. *Communications of the ACM*, 40(3): 88–89, 1997.

[3] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720, 1998.

[4] D. Billsus and M. Pazzani. A personal news agent that talks, learns and explains. *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 268–275, 2001.

[5] M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. *Proceedings of the International Conference on Intelligent User Interfaces*, pages 33–40, 2001.

[6] C. Drummond, R. Holte, and D. Ionescu. Accelerating browsing by automatically inferring a user's search goal. *Proceedings of the Eighth Knowledge-Based Software Engineering Conference*, pages 160–167, 1993.

[7] M. T. Gervasio, W. Iba, and P. Langley. Learning user evaluation functions for adaptive scheduling assistance. *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 152–161, 1999.

[8] Global Industry Classification Standard. Available at http://www.spglobal.com/GICSIndexDocument.PDF.

[9] J. Goecks and J. Shavlik. Learning users' interests by unobtrusively observing their normal behavior. *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 129–132, 2000.

[10] J. Karat. Evolving the scope of user-centered design. *Communications of the ACM*, 40(7): 33–38, 1997.

[11] J. Konstan, B. Miller, D. Maltz., J. Herlocker, L. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3): 77–87, 1997.

[12] P. Langley. User modeling in adaptive interfaces. *Proceedings of the Seventh International Conference on User Modeling*, pages 357–370, 1999.

[13] C. Lanquillon. Information filtering in changing domains. *Proceedings of the Workshop on Machine Learning for Information Filtering*, pages 41–48, 1999.

[14] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. *Proceedings of the Nineteenth International Conference on Research and Development in Information Retrieval*,pages 298–306, 1996.

[15] H. Lieberman. Letizia: An agent that assists Web browsing. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* pages 924–929, 1995.

[16] A. M. Rashid, I. Albert, D. Cosley, S. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: Learning new user preferences in recommender systems. *Proceedings of the International Conference on Intelligent User Interfaces*, pages 127–134, 2002.

[17] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl. Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 345–354, 1998.

[18] R. Segal and J. Kephart. MailCat: An intelligent assistant for organizing e-mail. *Proceedings of the Third International Conference on Autonomous Agents*, pages 276–282, 1999.

[19] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating 'word of mouth'. *Proceedings of Conference on Human Factors in Computing Systems*, pages 210–217, 1995.

[20] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Third edition. Addison-Wesley, Reading, MA, 1998.