A Personal News Agent that Talks, Learns and Explains

Daniel Billsus and Michael J. Pazzani
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697
+1 (949) 824-3491
dbillsus@ics.uci.edu, pazzani@ics.uci.edu

ABSTRACT

Most work on intelligent information agents has thus far focused on systems that are accessible through the World Wide Web. As demanding schedules prohibit people from continuous access to their computers, there is a clear demand for information systems that do not require workstation access or graphical user interfaces. We present a personal news agent that is designed to become part of an intelligent, IP-enabled radio, which uses synthesized speech to read news stories to a user. Based on voice feedback from the user, the system automatically adapts to the user's preferences and interests. In addition to time-coded feedback, we explore two components of the system that facilitate the automated induction of accurate interest profiles. First, we motivate the use of a multi-strategy machine learning approach that allows for the induction of user models that consist of separate models for long-term and short-term interests. Second, we investigate the use of "concept feedback", a novel form of user feedback that is based on our agent's capability to construct explanations for the reasons that have led to a specific classification. Users can then critique these explanations which, from a machine learning perspective, allows for more direct changes to an induced concept than through the inclusion of additional training examples. We evaluate the proposed algorithms on user data collected with a prototype of our system, and assess the performance contributions of the system's individual components.

Keywords

Information agents, machine learning, user modeling, human-computer interaction.

1. INTRODUCTION

Research on intelligent information agents, assistants that locate and retrieve information on behalf of their users, has recently attracted much attention. Most of this work has focused on agents accessible through the World Wide Web, and research in this field has not yet led to the development of interfaces for software agents that do not require access to a computer workstation. However, there is a clear demand for such information systems, as demanding schedules prohibit people from continuous computer access. Personalized information delivery has not yet made its way past the World Wide Web today.

Autonomous Agents '99 Seattle WA USA

The following example motivates the need for personalized information delivery outside of the World Wide Web. Users A and B spend a large portion of their day driving. They can listen to the radio in their cars, but do not have access to any medium that focuses on information specific to their interests. User A is primarily interested in business news in order to follow the current stock market, but he must listen to much unrelated information. To locate information, he must switch news channels and pay attention to the time at which broadcasts start. Similarly, user B must switch news channels to locate information on his interest in local sports.

One of our goals is to develop a foundation for intelligent information agents that do not require workstation access. Specifically, we envision an Internet Protocol (IP) enabled portable device that reads news stories to a user via a speech synthesizer. Based on feedback from the user, the system will automatically adapt to the user's preferences and assist in navigating through large information spaces. An interesting aspect of this device is that users can interrupt the synthesizer at any point, resulting in time-coded feedback that is not easily available in text-based applications. A long-term goal based on this research is the development of an intelligent, IP-enabled car radio that retrieves information from the Internet, learns about the driver's interests, and presents information in a personalized way. In general, we see IP-enabled appliances and the broadcast of textual information as an interesting alternative to traditional radio broadcasts. IP-enabled devices, e.g. radios, could take advantage of the lower bandwidth requirements of text vs. audio information, and a large amount of information could be transferred to these devices in a matter of seconds. Furthermore, textual representations open up ways to process, evaluate and recommend information, leading to personalized information access without the need for access to computer workstations. We believe that novel information retrieval and machine learning algorithms, specifically geared towards the characteristics of this kind of human-computer interaction will be needed to achieve the desired functionality.

This paper describes the design and features of an information agent that we currently use as a testbed to collect user data and evaluate our proposed algorithms. In particular, we describe how synthesized speech and time-coded feedback differ from more "traditional" relevance feedback approaches for written text. We then explain the induction of a detailed user model, and describe our agent's ability to revise its induced model through concept feedback. We evaluate the proposed algorithms on user data

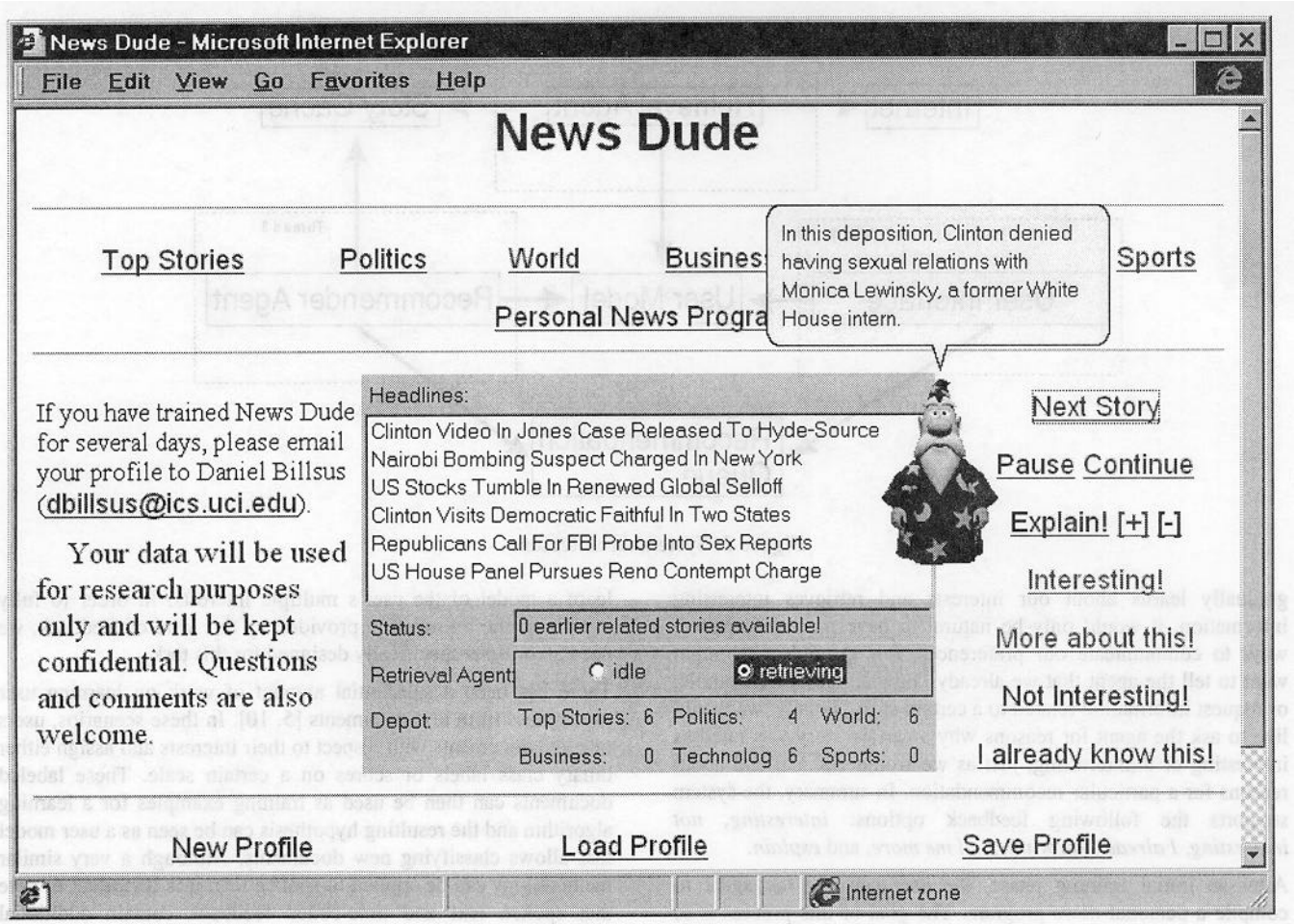

Figure 1: News Dude User Interface

collected with our agent, and assess the individual performance contributions of the system's components.

2. SYSTEM OVERVIEW

Building an agent that learns about a user's interests in daily news stories poses several challenges. Traditional Information Retrieval (IR) approaches are not directly applicable to this problem setting, because they assume the user has a specific, well-defined information need. In our setting, however, this is not the case. If at all, the user's query could be phrased as: "What is new in the world that I do not yet know about, but should know?'. Computing satisfactory results for such a query is non-trivial. The difficulty stems from the range of topics that could interest the user, and the user's changing interest in these topics. We must also take into account that it is the novelty of a story that makes it interesting. Even though a certain topic might match a user's interests perfectly, the user will not be interested in the story if it has been heard before. Therefore, we need to build a system that acquires a model of a user's multiple interests, is flexible enough to account for rapid interest changes, and keeps track of information the user knows.

We have implemented a Java Applet that uses Microsoft's Agent library to display an animated character, named News Dude, that reads news stories to the user. The Applet requires Microsoft's Internet Explorer 4.0 or newer, and is publicly accessible at http://www.ics.uci.edu/~dbillsus/NewsDude/. All functions can be accessed through voice commands, or the web-based user interface (see Figure 1). Although our ultimate goal is to work towards a speech-driven agent that does not require graphical user interfaces, we use the web as a medium that allows us to make the system available to a large user base for data collection and testing purposes. Furthermore, we believe that there is a variety of useful applications for speech-driven agent technology for the web. For example, a talking news-agent that reacts to voice commands could prove useful for the visually impaired.

Currently, the agent provides access to stories from six different news channels: Top Stories, Politics, World, Business, Technology and Sports. When the user selects a news channel, the Applet connects to a news site on the Internet *(Yahoo!News)* and starts to download stories. Since the Applet is multi-threaded, stories continue downloading in the background while the synthesizer is reading, which typically allows filling a queue of stories to be read without any waiting time. The user can interrupt the synthesizer at any point and provide feedback for the story being read. One of the design goals for our system was to provide a variety of feedback options that go beyond the commonly used interesting/uninteresting rating options. If we consider an intelligent information agent to be a personal assistant that

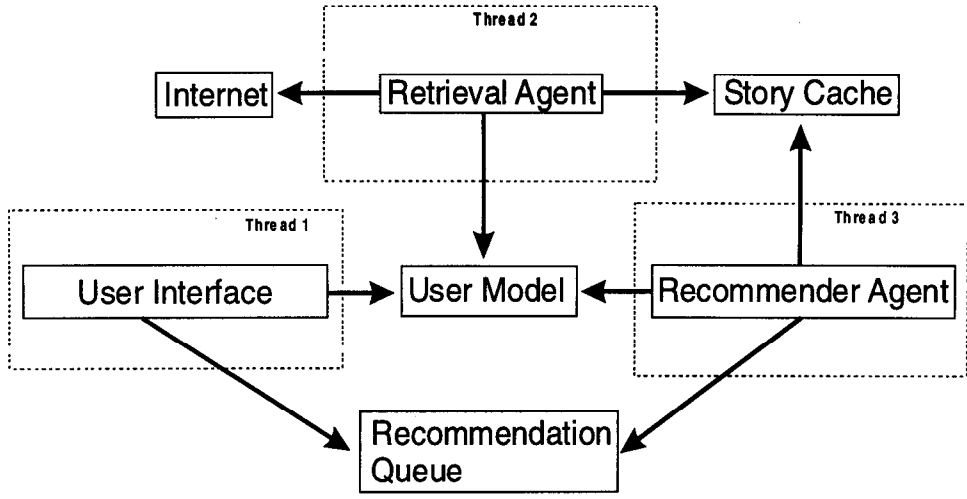

Figure 2: System Architecture

gradually learns about our interests and retrieves interesting information, it would only be natural to have more informative ways to communicate our preferences. For example, we might want to tell the agent that we already know about a certain topic, or request information related to a certain story. Finally, we would like to ask the agent for reasons why a certain story was rated as interesting or uninteresting, just as we would ask a friend about reasons for a particular recommendation. In summary, the system supports the following feedback options: *interesting, not interesting, I already know this, tell me more*, and *explain.*

After an initial training phase, the user can ask the agent to compile a personal news program. The goal of this process is to compute a sequence of news stories ordered with respect to the user's interest model. Figure 2 shows the three system components: retrieval agent, recommender agent and user interface. Each individual component runs in its own thread, to allow retrieving and recommending in the background while the user listens to stories. The retrieval agent's task is to connect to a news site on the Internet, download the latest news stories and insert them into the local story cache, which is used to queue all stories that are waiting to receive a relevance score. The recommender agent takes stories out of the story cache, uses the current user model to compute a relevance score and inserts the story into the sorted recommendation queue. When the user requests a new story, the user interface will take the top element out of the recommendation queue, and read it to the user. The user can then provide feedback, which is used to update the current user model. This results in a system that maintains an ordered sequence of news stories, inserts new stories into this sequence as soon as they are evaluated, and always presents the currently highest-ranked story to the user.

3. TIME-CODED FEEDBACK

Since our agent is designed to operate in environments where graphical user interfaces are not practical, voice-based operation is a natural alternative. News stories are read to the user through a speech synthesizer and the user can interrupt the synthesizer at any point and provide different forms of feedback, as described in the previous section. The agent's goal is to use this feedback to learn a model of the user's multiple interests. In order to fully exploit all the information provided by this kind of feedback, we need algorithms specifically designed for this task.

There has been a substantial amount of work on learning user preferences from text documents [5, 10]. In these scenarios, users rate text documents with respect to their interests and assign either binary class labels or scores on a certain scale. These labeled documents can then be used as training examples for a learning algorithm and the resulting hypothesis can be seen as a user model that allows classifying new documents. Although a very similar methodology can be applied to spoken text, it is important to note that spoken text and time-coded feedback contain additional information that might facilitate the learning and classification process. We believe that the point in time a user provides feedback or interrupts a news story is informative and should be incorporated into classification algorithms.

Time-coded feedback is incorporated into the classification process of our current system by using the amount of time a user has listened to a story as implicit feedback. The underlying intuition is that users will listen longer to stories they consider interesting than to stories they are not interested in, and we would like to capture this form of evidence. In our current implementation we convert a user's binary rating (interesting vs. not interesting) into a more fine-grained scale. Scores are computed according to the following rules. Let *pl* be the proportion of a story the user has heard.

If story was rated as uninteresting, $score = 0.3 * pl$

If story was rated as interesting, $score = 0.7 + 0.3 * pl$

If user asked for more information, $score = 1.0$

This scheme assures that stories rated as not interesting always receive lower scores than stories rated as interesting, but allows for differentiation between different levels of ratings without requiring any extra work by the user. Similar techniques for implicit feedback have previously been studied [7]. Additional techniques for the use of time-coded feedback are currently under investigation and are briefly summarized in Section 7.

4. LEARNING A USER MODEL

The specific design of our agent's user model is motivated by a number of observations and requirements. First, the model must be capable of representing a user's multiple interests in different topics. Second, the model must be flexible enough to adapt to a user's changing interests reasonably quickly, even after a long preceding training period. Third, the model should take into account that a user's information needs change as a direct result of interaction with information [2]. Surprisingly, this aspect has received virtually no attention in the IR community. For our application, we take into account the stories the user has recently heard, to avoid presenting the same information twice.

The above requirements led to the development of a multi-strategy learning approach that learns two separate user-models: one represents the user's short-term interests, the other represents the user's long-term interests. The need for two separate models can be further substantiated by the specific task at hand, i.e. classifying news stories. Users typically want to track different "threads" of ongoing recent events - a task that requires short-term information about recent events. In addition, users have general news preferences, and modeling these general preferences may prove useful for deciding if a new story, which is not related to a recent rated event, would interest the user. In the following sections we describe the two models individually, motivate why they conform to the above requirements, and describe how they can work together to form a comprehensive user model appropriate for our purposes.

4.1 Modeling Short-Term Interests

The purpose of the short-term model is two-fold. First, it should contain information about recently rated events, so that stories which belong to the same threads of events can be identified. Second, it should allow for identification of stories that the user already knows. A natural choice to achieve the desired functionality is the nearest neighbor algorithm (NN). The NN algorithm simply stores all its training examples, in our case rated news stories, in memory. In order to classify a new, unlabeled instance, the algorithm compares it to all stored instances given some defined similarity measure, and determines the "nearest neighbor" or the k nearest neighbors. The class label assigned to the new instance can then be derived from the class labels of the nearest neighbors. The utility of the NN algorithm has previously been explored in other text classification applications [1, 12].

To apply the algorithm to natural language text, we must define a similarity measure that quantifies the similarity between two text documents. This is a well-studied problem in Information Retrieval, and we rely on a commonly used document representation and an associated similarity measure. We convert news stories to *TF-IDF* vectors (term-frequency / inverse-document-frequency), and use the cosine similarity measure to quantify the similarity of two vectors [11].

Each rated story is converted to its *TF-IDF* representation and then stored in the user model. A score prediction for a new story is then computed as follows. All stories that are closer than a threshold *t_min* to the story to be classified become voting stories. The predicted score is then computed as the weighted average over all the voting stories' scores, where the weight is the similarity between a voting story and the new story. If one of the voters is closer than threshold *t_max* to the new story, the story is labeled as known, and its computed score is multiplied by a constant $k << 1.0$, because the system assumes that the user has already heard about the event reported in the story. If a story does not have any voters, the story cannot be classified by the short-term model at all, and is passed on to the long-term model (see Section 4.2).

The nearest neighbor based short-term model satisfies our requirements that a user model be able to represent a user's multiple interests, and it can quickly adapt to a user's novel interests. The main advantage of the nearest neighbor approach is that only a single story of a new topic is needed to allow the algorithm to identify future follow-up stories from the same story thread. The "tracking" abilities of the nearest neighbor algorithm have also recently been explored by other researchers in a similar project [1]. In contrast, most other learning algorithms would require a large number of training examples to identify a strong pattern.

4.2 Modeling Long-Term Interests

The purpose of the long-term user model is to model a user's general preferences for news stories and compute predictions for stories that could not be classified by the short-term model. To achieve this we selected a probabilistic learning algorithm, the naïve Bayesian classifier [4]. Naïve Bayes has been shown to perform competitively with more complex algorithms and has become an increasingly popular algorithm in text classification applications [9, 10].

We represent news stories as Boolean feature vectors, where each feature indicates the presence or absence of a word. Not all the words that appear in news stories are used as features. Since it is our explicit goal to model a user's general preferences, we provide the algorithm with background knowledge by hand-selecting a set of domain specific features, i. e. words that are likely to be indicators for commonly recurring themes in daily news stories. Approximately 200 words were selected, ranging from countries to crime, disaster, politics, technology, business and sport related terms. Making the "naïve" assumption that features, here words, are independent given the class label (interesting vs. not interesting), the probability of a story belonging to class j given its feature values, $p(class_j | f_1, f_2, \ldots f_n)$ is proportional to:

$$p(class_j)\prod_i^n p(f_i \mid class_j)$$

where $p(class_j)$ and $p(f_i \mid class_j)$ can be easily estimated from training data. Specifically, we use the multi-variate Bernoulli event model formulation of naïve Bayes [9], and compute Bayes-optimal estimates of $p(class_j)$ and $p(f_i \mid class_j)$ by straightforward counting of word and class occurrences in the training data. We use Laplace smoothing to prevent zero probabilities for infrequently occurring words. A news story to be classified can thus be labeled with its probability of belonging to the interesting class.

Most applications of the multi-variate Bernoulli formulation of naïve Bayes consider both the presence and absence of words in text documents as evidence in the probability computation. We restrict the evidence used to the presence of words, similar to a naïve Bayes model proposed by Maron [8]. This results in a more conservative classifier that requires examples classified as class c to be similar to other examples in class c.

Finally, we would like to prevent the long-term model from classifying stories that do not contain a sufficient number of features that are indicators for class membership. More formally, we require the story to contain at least n features for which $p(f \mid interesting) > p(f \mid not\ interesting)$ in order to allow a classification as interesting, and likewise, at least n features for which $p(f \mid interesting) < p(f \mid not\ interesting)$ in order to allow a classification as not interesting. In our current implementation we set n to 3, which means a story must contain at least 3 terms that are all indicators for the same class.

4.3 A Hybrid User Model

Using a hybrid user model consisting of both a short-term and long-term model of the user's interests, a previously unseen news story is classified as follows:

If story can be classified by short-term model

{

score = weighted average over nearest neighbors

If story is too close to known story

*score = score * SMALL_CONSTANT*

}

Else

If story can be classified by long-term model

score = probability estimated by naïve Bayes

Else

score = DEFAULT_SCORE

In summary, the approach tries to use the short-term model first, because it allows the user to track news threads that have previously been rated, and it can label stories as already known. If the story cannot be classified with the short-term model, the long-term model is used. If the long-term model decides that the story does not contain sufficient evidence to be classified, a default score is assigned. In the current implementation we set the default score to 0.3, so that stories that cannot be reliably classified do not appear too high in the recommendation queue, but still receive a higher score than stories that are classified as not interesting.

5. EXPLAINING RECOMMENDATIONS

One of the goals of our work is to support expressive forms of dialog between users and the agent that facilitate the acquisition of a precise user model. This is achieved through the various feedback options described in previous sections, as well as our agent's ability to explain reasons for its recommendations. Systems that can form explanations for their decisions have a long history in Artificial Intelligence, especially in the context of Expert Systems [3]. Explanations can help a user understand the system's reasoning and provide a way to verify the validity of a decision.

Explanation capabilities have not received much attention in the context of Information Retrieval and software agents. Typically, a system forms a model of the user's information need, uses the model to retrieve relevant items and then presents these items to the user. While it is common for Information Retrieval systems to quantify the relevance of located information in the form of a single score, the user typically does not get any form of explanation or justification for the computed score.

We believe that an explanation for the main reasons that led to a specific score might be useful for several reasons. First, an explanation can help the user decide if he wants to inspect a located item further. This might be especially useful in the audio framework of our agent, where users cannot skim located items in order to briefly assess their relevance. If requested, a short explanation can help the user decide if he wants to listen to a story. Second, explanations provide direct insight into the induced user model. This allows the user to assess whether the specific aspect of the user model that led to a certain recommendation is useful for finding relevant information. Therefore, our agent allows the user to critique the formed explanations in order to make direct changes to the induced model. We refer to this form of interaction as "concept feedback".

5.1 Explanation Construction

The system's recommendations are based on scores computed by the hybrid user model consisting of separate short-term and long-term models as described in Section 4. As a consequence, different forms of explanations are used to summarize reasons for a story's relevance.

If the story was similar to a previously rated story and therefore classified by the short-term model, the explanation is based on proximity to this previously rated story. The agent retrieves the headline of the closest story in the short-term model that received the same class label as the story whose explanation is to be constructed. The retrieved headline can then be used to construct explanations of the following form:

Explanation Template 1: *"This story received a* ***[high \| low]*** *relevance score, because you told me earlier that you were* ***[not]*** *interested in* ***[closest_headline]"***

Likewise, if the system assigns a low relevance score because it assumes the user is already familiar with the story's content, a proximity-based explanation of the following form is constructed.

Explanation Template 2: *"I think you already know about this, because I told you earlier that* ***[closest_headline]"***.

In contrast, if the story was classified by the long-term model to belong to class c, the system forms an explanation using the words that influenced the class decision most. For each feature f appearing in the story, we compute $influence_f = log\ [p(f \mid c) / p(f \mid not\ c)]$, and determine the n words with the highest influence value. In our current implementation we set n to 3, and construct explanations of the following form.

Explanation Template 3: *"This story received a* ***[high \| low]*** *relevance score, because it contains the words f1, f2 and f3."*

Finally, if the story received a default score, the system explains that *"the story received a default score, because it did not seem to be related to any previously rated story, and did not contain enough informative words that would allow classification"*.

5.2 Concept Feedback

Users respond to the system's explanation with a binary rating indicating whether the line of reasoning expressed in the explanation matches their preferences, and is thus useful or not useful for future recommendations. Since the system's explanations correspond directly to specific "concepts" represented in the user model, this form of feedback allows for direct changes to the induced model. Concept feedback is therefore significantly different from more traditional relevance feedback approaches [11], where ratings refer to the relevance of

items, instead of the concepts that were used to classify items. The underlying intuition for the use of concept feedback is that it could lead to three potential benefits. First, it might lead to user models that reflect a user's preferences more accurately. Second, it could lead to a reduction of training data needed to achieve a certain accuracy. Third, it could lead to more flexible user models, which is very useful in our domain where we assume a user's preferences to be unstable.

Depending on the explanation template used to form an explanation, user feedback affects the learned model in different ways.

For the proximity-based explanation **template 1**, we take the similar story used in the explanation out of the short-term model if the explanation received negative feedback, or add it a second time if the explanation received positive feedback. Removing the story from the short-term model prevents it from contributing to further misclassifications. This is especially useful in cases where users decide they no longer want to follow a thread of stories they have previously rated as interesting. In contrast, adding the story to the model again allows the story to vote multiple times in future recommendations.

If the user provides positive feedback for an explanation formed with the proximity-based explanation **template 2**, i.e. in cases where the system assumes that the user already knows a story, the current content of the user model remains unchanged. However, if the user provides negative feedback for this explanation, we increase the threshold *t_max* (see Section 4.1) by a small constant, so that future stories are required to be more similar to rated stories, before they are classified as already known. Likewise, *t_max* is reduced if the system classifies a news story as interesting, and the user indicates that the story has been heard before.

The agent reacts to feedback for explanations formed with explanation **template 3** by constructing an additional, "artificial" training example. Since this explanation is used for stories that were classified based on a set of words that indicate class membership according to the long-term model, the newly created training example consists simply of the words f_1, f_2 and f_3 that were used in the explanation. This new training example is then added to both, the long-term and short-term model, and its class label depends on the user's feedback. The effect of adding the new example to the long-term model is essentially that the frequency counters used to determine $p(f_n|class)$ are updated, making future misclassifications based on the same words less likely. However, adding the example to the short-term model has a more dramatic effect. Consider the case where the system explains a high relevance score with the presence of the words "Internet, Web and Software". If the user provides positive feedback for this explanation, the newly created positive training example containing "Internet, Web and Software" assures that future stories containing these words will be detected by the short-term model, causing the story to have a high position in the recommendation queue, regardless of the remaining information in the story. Likewise, if the user provides negative feedback for a story that received a high relevance score due to the words "President, Congress and Affair", a negative training example will be added which causes other stories that contain these words to be filtered out by the short-term model.

6. EVALUATION

In order to evaluate the recommendation performance of our agent and to assess the relative performance contributions of its individual components, we used the web-based agent prototype to collect user data. Ten users trained the system on a daily basis over a period ranging from 4 to 8 days, resulting in about 3,000 total rated news stories, i.e. on average 300 stories per user. While this amount of data might not lead to overall performance estimates that generalize to other users or different collection dates, it allows us to analyze the relative performance of the system's individual components.

Evaluating the agent's performance is difficult for several reasons. First, standard evaluation methodologies commonly used in the machine learning literature, for example n-fold cross-validation, are not applicable to this scenario. This is mainly due to the chronological order of the training examples, which cannot be presented to the learning algorithm in random order, without skewing results. Second, if we measure the agent's performance on a daily basis, we not only measure the effects of the agent's updated user model, but also of the changing distribution of news stories. Finally, we are trying to approximate a model of user interests that can be assumed to be neither static nor consistent. A user going through the same list of stories at a later time might assign different labels.

We chose to evaluate the agent's performance as follows. We divided each user's data into separate training sessions, corresponding to the user's use of the system, i.e. typically one training session per day. We started to train the algorithm with all rated examples from the first training session, and compared its predictions for class labels of stories from the second training session to the user's ratings. We then incremented the training set session by session and measured the agent's performance on the following session. Finally, we averaged the results over all users. This methodology models the way the system is used realistically, because all training data available up to a certain day was used to classify stories.

In addition to the system's classification accuracy, i. e. the proportion of correctly classified news stories, we used common Information Retrieval performance measures, precision, recall and F_1 to evaluate the system. It is important to evaluate precision and recall in conjunction, because it is easy to optimize either one separately. However, for a classifier to be useful for our purposes we demand that it be precise as well as have high recall. In order to quantify this with a single measure, Lewis and Gale [6] proposed the *F-measure*, a weighted combination of precision and recall that produces scores ranging from 0 to 1. Here we assign equal importance to precision and recall, resulting in the following definition for F_1:

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Figure 3 summarizes the system's performance averaged over all users, showing a rapid increase of classification performance during the first three training sessions. Results for more training sessions are only available for a subset of the users and therefore cannot be presented in one plot. However, results for users that collected data of up to 8 training sessions revealed that performance seems to increase rapidly during the first few training sessions and then starts to fluctuate as a result of changing

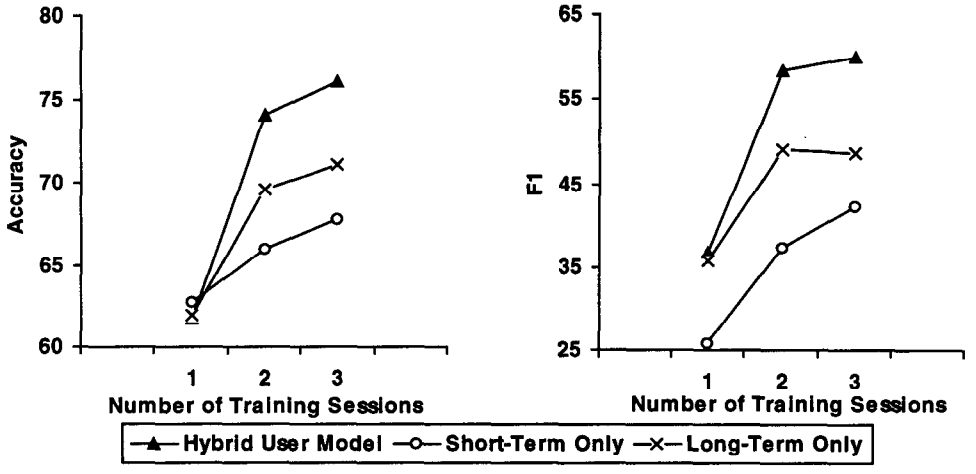

Figure 3: Overall System Performance

distributions of daily news stories. Figure 3 also shows the relative performance of the two user model components. As expected, the hybrid approach combining a short-term and long-term user model performs better than each individual approach with respect to both classification accuracy and the F_1 measure. Further inspection of the achieved performance revealed that the short-term model tends to have high precision, but low recall. In contrast, the long-term model has higher recall than the short-term model, but lower precision. Combining both models allows taking advantage of both models' strengths, resulting in higher F_1 values as well as overall classification accuracy.

In order to evaluate the contribution of time-coded feedback, we converted the time-coded scores to Boolean ratings and measured the resulting performance difference. Using classification accuracy and F_1, no significant difference could be observed. This is not surprising if we take into account that these measures only change if the time-coded information causes individual news stories to change their class membership. However, time-coded scores did significantly change the relative order of stories in the recommendation queue. To assess the magnitude of this effect, we measured the system's precision at the top 5 suggestions. The plot shown in Figure 4 summarizes these results and shows that time-coded scores led to a precision increase of about 8%.

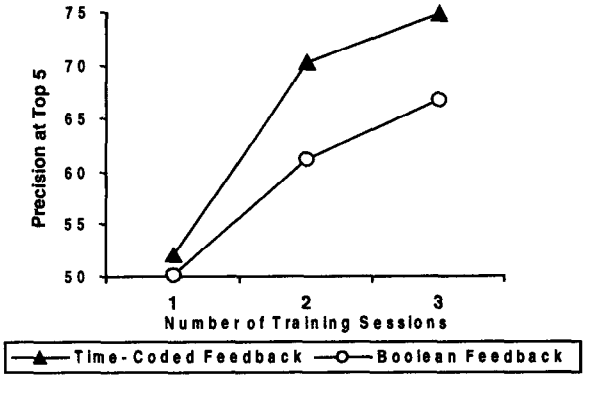

Figure 4: Effect of Time-Coded Feedback

Finally, we evaluated the effects of concept feedback on the overall system performance. Since performance differences depend on the number of concept feedback interactions provided by users, it is difficult to visualize results in the form of learning curves. Therefore, we used all training data provided by each user to predict class labels for the stories from the user's final training session. We then measured the resulting performance increase by adding all concept feedback interactions and averaged the results over all users. This caused accuracy to increase from 72.5% to 77.1%, and F_1 to increase from 60.1% to 64.7%.

7. FUTURE WORK

By taking advantage of the length of time a user listens to a news story, we have only begun to explore the potential of time-coded feedback. We are currently exploring approaches where we modify the text representation of a news story by reweighting the portion of the story to which the user has actually listened. Consider the case where a user interrupts a news story after the first 10 words read and decides to label the message as uninteresting. Clearly, we should treat the first 10 words, i.e. the portion of the story the user actually listened to, differently from the remaining text of the story. Since the user was able to reach a decision after the first 10 words, we assume that one or more strong indicators for class membership are among these words. Since we represent news stories in the form of *TF-IDF* vectors, it is possible to artificially modify those weights in order to assign more weight to words to which the user has listened.

We are currently implementing a user interface specifically aimed at visually impaired users. While intelligent information filtering techniques have been used to personalize access to various types of information available on the Internet, it is surprising that the potential benefits of these techniques to visually impaired users have thus far not been explored. Since visually impaired users cannot easily skim retrieved information to quickly assess potential relevance, information agents that know about user preferences and interests could prove invaluable.

8. ACKNOWLEDGMENTS

We would like to thank Daimler-Benz and Sun Microsystems, Inc. for their generous support. We also thank the users who trained our system for assistance with this research project.

9. REFERENCES

[1] Allan, J., Carbonell, J.G., Doddington, G., Yamron, J. and Yang, Y. Topic Detection and Tracking Pilot Study Final Report, *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop,* 1998, Lansdowne, Virginia.

[2] Belkin, N. User Modeling in Information Retrieval, *http://www.scils.rutgers.edu/~belkin/um97oh/,* Tutorial Overheads, *Sixth International Conference on User Modeling*, 1997, Chia Laguna, Sardinia.

[3] Cawsey, A. *Explanation and Interaction.* Cambridge, MA: The MIT Press, 1992.

[4] Duda, R., and Hart, P. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[5] Lang, K. NewsWeeder: Learning to filter news. *Proceedings of the Twelfth International Conference on Machine Learning,* Lake Tahoe, CA, 1995, 331–339.

[6] Lewis, D. and Gale, W.A. A sequential algorithm for training text classifiers. *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 1994, London, Springer Verlag.

[7] Lieberman, H. Letizia: An agent that assists web browsing. *Proceedings of the International Joint Conference on Artificial Intelligence*, Montreal, August 1995, 924–929.

[8] Maron, M. Automatic Indexing: An Experimental Inquiry. *Journal of the Association for Computing Machinery*, 1961, 8:404-417.

[9] McCallum, A. and Nigam, K. A Comparison of Event Models for Naïve Bayes Text Classification. *AAAI/ICML-98 Workshop on Learning for Text Categorization.* Technical Report WS-98-05, 1998, AAAI Press.

[10] Pazzani M., and Billsus, D. Learning and Revising User Profiles: The identification of interesting web sites. *Machine Learning 27*, 1997, 313-331.

[11] Salton, G. *Automatic Text Processing.* Addison-Wesley, 1989.

[12] Yang, Y. An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval Journal*, (to appear).